Oracle® Fusion Middleware

Forms Services Deployment Guide

11*g* Release 1 (11.1.1)

E10240-07

June 2012

ORACLE®

Oracle Fusion Middleware Forms Services Deployment Guide 11*g* Release 1 (11.1.1)

E10240-07

Primary Author: Swati Thacker

Contributors: Ananth Satyanarayana, Vinay Agarwal, Suvarna Balachandra, Hemant Bansal, Ramesh Gurubhadraiah, Laiju Mathew, Gururaja Padakandla, Opendro Singh, Ashish Tyagi, Sudarshan Upadhya, Syed Nisar Ahmed, Dhiraj Madan, James Amalraj, Phil Kuhn, Arthur Housinger, Rubik Sadeghi, Naseer Syed, Emerson deLaubenfels, Grant Ronald

Contents

4 Configuring and Managing Forms Services

5 Using Oracle Forms Services with the HTTP Listener and Oracle WebLogic Server

9 Using Forms Services with Oracle Single Sign-On

10 Configuring and Managing Java Virtual Machines

11 Forms Services Security Overview

12 Tracing and Diagnostics

Preface

Intended Audience

This manual is intended for software developers who are interested in deploying Oracle Forms applications to the Web with Oracle Fusion Middleware.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/us/corporate/accessibility/index.html.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following manuals:

- *Oracle Fusion Middleware Release Notes for Linux x86*
- *Oracle Fusion Middleware Release Notes for Microsoft Windows*
- *Oracle Forms Upgrading Oracle Forms 6i* to Oracle Forms 11*g*
- Oracle Fusion Middleware Library on OTN
- Oracle Forms Builder Online Help, available from the Help menu in Oracle Forms Developer.

In addition, you will find white papers and other resources at http://www.oracle.com/technology/products/forms/.

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.

1 What's New in Oracle Forms Services

This chapter describes the features and improvements in 11*g* Release 1 of Oracle Fusion Middleware Forms Services.

1.1 JavaScript Integration

Use JavaScript integration that is now available in Oracle Forms 11*g* to have JavaScript call into your Forms applet, or have your Forms applet execute JavaScript.

For more information, see Chapter 6, "Oracle Forms and JavaScript Integration."

1.2 Enhanced Java Support

You can extend Pluggable Java Components (PJC) to raise events in Oracle Forms Services.

For more information, see Section 7, "Enhanced Java Support."

1.3 Support for Server-Side Events

Oracle Forms Services supports external events from outside of Forms Services. An event can be raised by the database or by BPEL through Oracle Advanced Queueing.

For more information, see Chapter 8, "Working with Server Events."

1.4 Proxy User Support

In this release of Oracle Forms, Forms developers can choose to have users connect to the database as a proxy user, a single database user is a user with connection privileges only, adding security and simplifying user management in the process while maintaining automatic database auditing.

For more information, see Section 9.6, "Enabling and Configuring Proxy Users."

1.5 PL/SQL Tracing

In Oracle Forms Services 11*g*, you can enable logging of the names and parameters for called PL/SQL Procedures and functions, then view the output in Forms Trace.

For more information, see Chapter 12, "Tracing and Diagnostics."

1.6 Integration with Oracle Diagnostics and Logging (ODL)

Oracle Diagnostic logging (ODL) is a feature of Oracle Fusion Middleware that extends the J2SE logging framework. ODL makes it easier to diagnose problems and manage log files in Oracle Fusion Middleware.

For more information, see Section 12.6, "Taking Advantage of Oracle Diagnostics and Logging Tools."

2

Introduction to Oracle Forms Services

This chapter introduces Oracle Forms. It provides an overview of the development and deployment environment for Oracle Forms, and provides references where you can find more information on associated components in Oracle Fusion Middleware.

This chapter contains the following sections:

2.1 Oracle Forms

Oracle Forms is a component of Oracle Fusion Middleware. Oracle Forms is used to develop and deploy Forms applications. The Forms applications provide a user interface to access Oracle Database in an efficient and tightly-coupled way. The applications can be integrated with Java and web services to take advantage of service oriented architectures (SOA).

Oracle Forms includes the following:

- Oracle Forms Developer, used to develop and compile Forms applications.
- Oracle Forms Services, a server component, used to deploy the applications.

2.1.1 Oracle Forms Developer

Oracle Forms Developer is used to develop a form that can access an Oracle database and present the data. Wizards and utilities are provided to speed up application development. The source form (*.fmb) is created and compiled into an "executable" (*.fmx). The Forms application is run (interpreted) by the Forms Runtime process.

For more information about the Oracle Forms Developer, refer to the following documentation:

- Oracle Forms Builder Online Help, which is accessible from Oracle Forms Builder, provides information on how to use Oracle Forms Developer to develop and compile Forms applications.
- *Upgrading Oracle Forms 6i to Oracle Forms 11g*: describes obsolete features of Oracle Forms Developer and instructions for upgrading your Forms applications.

2.1.2 Oracle Forms Services

Oracle Forms Services is a comprehensive application framework optimized to deploy Forms applications in a multitiered environment. It takes advantage of the ease and accessibility of the Web and elevates it from a static information-publishing mechanism to an environment capable of supporting complex applications.

The Form applications that you design and develop in Oracle Forms Developer are deployed on Oracle Fusion Middleware. These applications run on the middle tier (see Figure 2–2). The user interface is presented on the client tier as a Java applet in the client's browser.

This guide describes the configuration files, and environment variables that can be used to customize deployment of Forms applications. It also provides information on performance, logging and monitoring your deployment. You can use Oracle Fusion Middleware Enterprise Manager Control to manage the configuration files, and environment variables, and monitor the deployment.

2.1.3 How Oracle Forms Services Launches a Forms Application

When a user first starts an Oracle Forms application by clicking a link to the application's URL, the baseHTML file is read by the Forms servlet. Any variables (%*variablename*%) in the baseHTML file are replaced with the appropriate parameter values specified in the `formsweb.cfg` file, and from query parameters in the URL request (if any).

You can easily modify the configuration files with Oracle Enterprise Manager Fusion Middleware Control as your needs change. Section 2.6, "Oracle Forms Services Architecture" describes the processes that are involved in deploying and running a typical Forms application.

2.2 Oracle Database

Oracle Database is the latest generation of RDBMS. Among the numerous capabilities are unlimited scalability and industry-leading reliability with Oracle Real Application Clusters; high availability technology including advancements in standby database technology (Oracle Data Guard); and built-in OLAP, data mining and Extract, Transform and Load (ETL) functions.

For more information on Oracle Database, refer to `http://www.oracle.com/technology/documentation/index.html`.

2.3 Oracle WebLogic Server

Oracle WebLogic Server 11*g* Release 1 is an application server for building and deploying enterprise Java EE applications with support for new features for lowering cost of operations, improving performance and supporting the Oracle applications portfolio.

Regardless of whether you want to create a staging, production, or testing environment, you begin by creating a WebLogic domain. A WebLogic domain includes instances of WebLogic Server, of which one is configured as an Administration Server. The Administration Server maintains configuration data for a domain. You can deploy your application on Administration Server but it is recommended to create a managed server and deploy your application in managed server. For more information on Oracle WebLogic Server, refer to *Oracle Fusion Middleware Introduction to Oracle WebLogic Server*.

During configuration, a managed server for Forms is created (WLS_FORMS). For more information on WLS_FORMS, refer to Section 5.1, "About the Oracle WebLogic Managed Server."

2.4 Oracle Fusion Middleware

Oracle Fusion Middleware includes Web servers, application servers, content management systems, and developer tools that provide complete support for development, deployment, and management of software applications. Among the components are Oracle Forms Services, Oracle WebLogic Server, and Oracle Enterprise Manager Fusion Middleware Control, which together provide the technology to fully realize the benefits of Internet computing.

You can manage and monitor Oracle Forms using Oracle Enterprise Manager Fusion Middleware Control.

For a complete overview, list of components, and conceptual information about Oracle Fusion Middleware, refer to the following manuals:

- *Oracle Fusion Middleware Concepts*
- *Oracle Fusion Middleware Administrator's Guide*

2.5 About Installing or Upgrading Oracle Forms

Oracle Forms is installed from the *Oracle Portal, Forms, Reports and Discoverer 11g (11.1.1.4.0) DVD*. In the installer, you can selectively configure any one of these products or all of them. For more information on installing Oracle Forms, refer to the following guides:

- *Oracle Fusion Middleware Installation Planning Guide*
- *Oracle Fusion Middleware Installation Guide for Oracle Portal, Forms, Reports and Discoverer*
- *Oracle Fusion Middleware Quick Installation Guide for Oracle Portal, Forms, Reports, and Discoverer*

For upgrade information, refer to the following documents:

- *Oracle Fusion Middleware Upgrade Planning Guide*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Portal, Forms, Reports, and Discoverer*

For information on upgrading Forms 6*i* to Oracle Forms 11*g*, see Chapter 13, "Upgrading to Oracle Forms Services 11g."

For information about changed or obsolete features, see the *Oracle Forms Upgrading Oracle Forms 6i to Oracle Forms 11g Guide*.

Note: After you have installed Oracle Forms, you can use the following options to save RAM in a development-only environment:

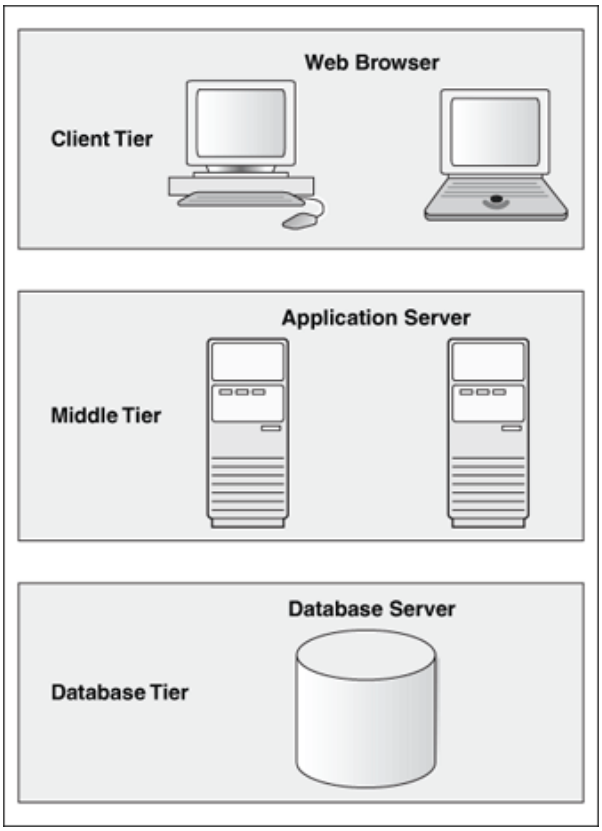
- You can choose to stop `WLS_REPORTS` (or other managed servers that may be running). To test Forms applications, only `WLS_FORMS` is required.
- By default, `formsapp` and `formsconfigmbeans` run on `WLS_FORMS`. You can retarget these applications to run on the Administration Server and stop the Forms managed server (`WLS_FORMS`).

2.6 Oracle Forms Services Architecture

Figure 2–1 shows the three-tier architecture that makes up Forms Services:

- The **client tier**, at the top of the image, contains the Web browser, where the application is displayed. In addition to the browser, Java Runtime Environment (JRE) and Java Plug-In (JPI) are required. For more information, see Appendix B, "Configuring Java Plug-ins" and `http://java.sun.com/reference/docs/`.
- The **middle tier,** in the center of the image, is the application server, where application logic and server software are stored.
- The **database tier**, in the lower portion of the image, is the database server, where database server software is stored.

Figure 2–1 Oracle Forms Services Architecture



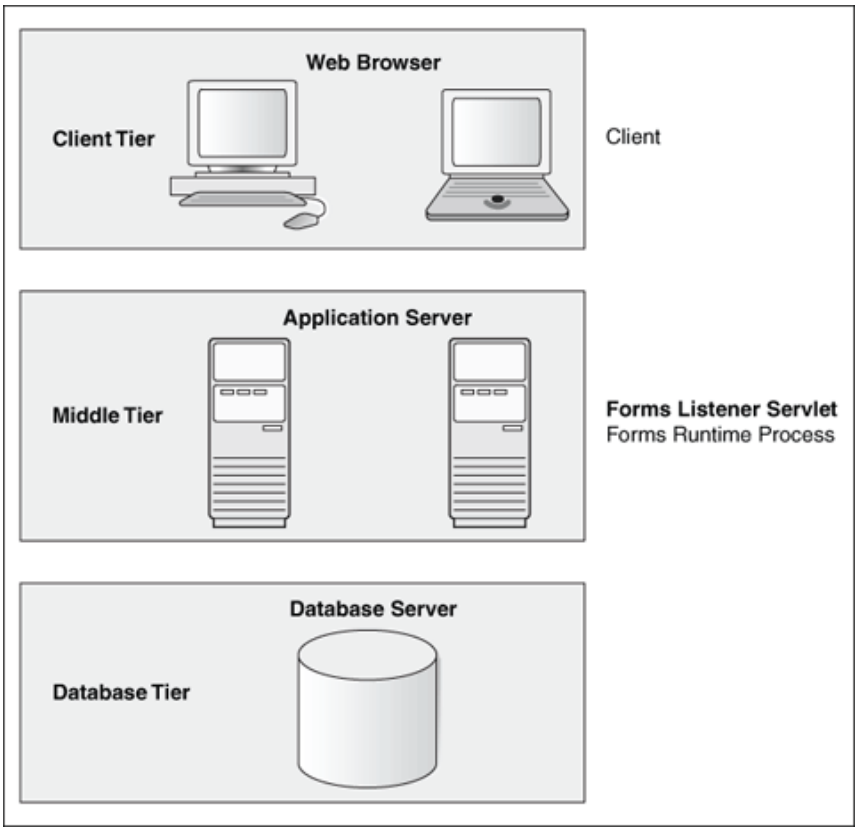
2.6.1 Oracle Forms Services Components

Oracle Forms Services is a middle-tier application framework for deploying complex, transactional forms applications to a network such as an intranet or the Internet. Developers build Forms applications with Forms Developer and deploy them with Forms Services. Developers can also take current applications that were previously deployed in client/server and move them to a three-tier architecture. Some minor changes in application code may be required when moving to a three-tier architecture.

As shown in Figure 2–2, the three-tier configuration for running a form consists of:

- The **Client**, at the top of the image, resides on the client tier
- The Forms Listener servlet, in the center of the image, resides on the middle tier
- The **Forms Runtime process**, also resides on the middle tier

Figure 2–2 Three-tier configuration for running a form



2.6.1.1 Forms Listener Servlet

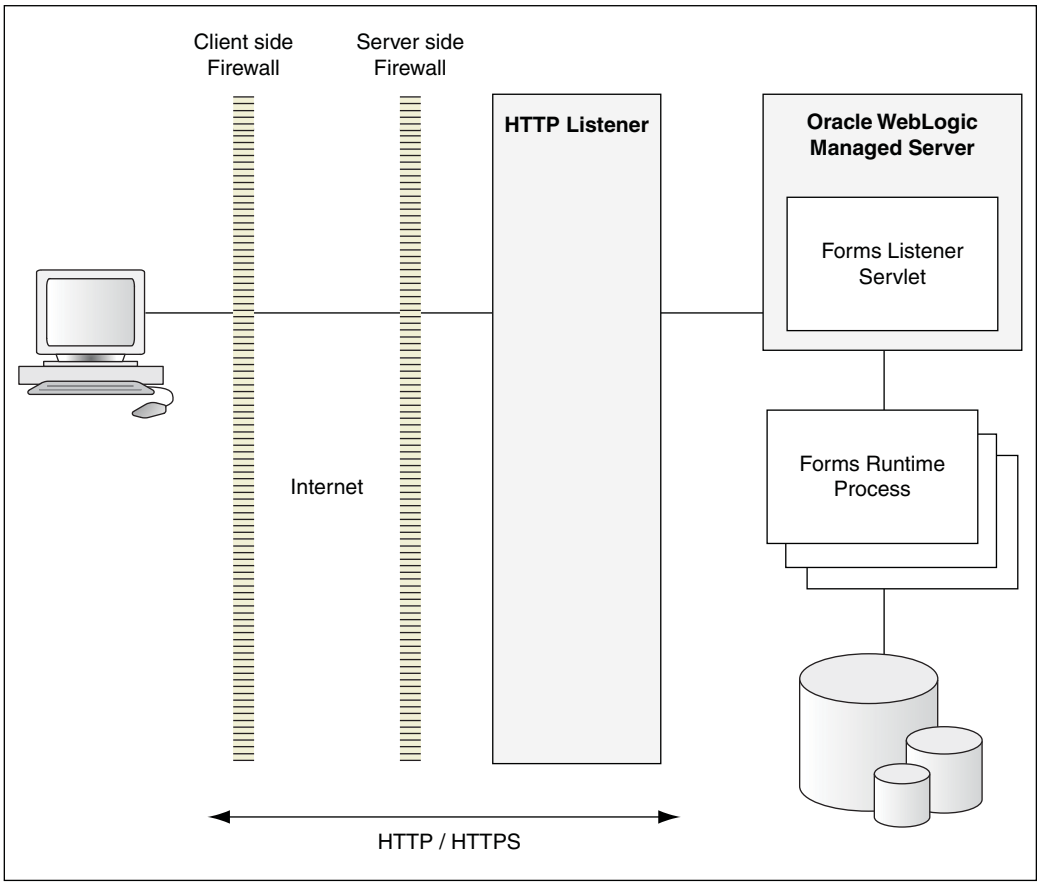
The Forms Listener servlet is a broker between the Java client and the Forms Runtime process. It takes connection requests from Java client processes and initiates a Forms Runtime process on their behalf.

Figure 2–3 illustrates how the client sends HTTP requests and receives HTTP responses from Forms Services. Oracle Forms Services uses the Forms Listener servlet to start, stop, and communicate with the Forms Runtime process. In this image, the client is to the left. In the center of the image, the HTTP Listener acts as the network endpoint for the client, keeping the other server computers and ports from being exposed at the firewall.

The Forms Runtime process, in the right side of the image, executes the code contained in a particular Forms application. The Forms Listener servlet manages the creation of a Forms Runtime process for each client and manages the network communications between the client and its associated Forms Runtime process.

Note: The Forms Listener servlet is configured for you during the Oracle Fusion Middleware installation process.

Figure 2–3 Architecture using the Forms Listener Servlet



2.6.1.2 Forms Runtime Process

The Forms Runtime process plays two roles: when it communicates with the **client browser**, it acts as a server by managing requests from client browsers and it sends metadata to the client to describe the user interface; when it is communicating with the **database server**, it acts as a client by querying the database server for requested data.

For each Oracle Forms session, there is one Oracle Forms Runtime process on the application server. This process is where Oracle Forms actually runs, and manages application logic and processing. It also manages the database connection; queries and updates data; runs any PL/SQL in the Form; executes triggers; and so on. It uses the same forms, menus, and library files that were used for running in client/server mode.

The Forms Runtime process also contains the Java Virtual Machine (JVM) to run Java in your application. As an optimization feature, the JVM is started if the Forms

application uses the Java Importer. In 10*g*, the JVM pooling feature is used only by the Java Importer. In 11*g*, Forms Runtime Process no longer creates a separate JVM when it calls Reports. Instead, if a JVM controller is configured for a form, the form can use the shared JVM when calling Reports. This results in a reduction of memory consumption, freeing more resources on the server. For more information about managing JVM usage and pooling, see Chapter 10, "Configuring and Managing Java Virtual Machines."

3

Basics of Deploying Oracle Forms Applications

This chapter describes how Forms Services run in Oracle Fusion Middleware, and describes the steps to deploy Forms applications. This chapter also describes the basic configuration files. After installation is completed, you can use the information in this chapter to change your initial configuration or make modifications as your needs change.

This chapter contains the following sections:

3.1 Oracle Forms Services in Action

This section describes how Forms Services run in Oracle Fusion Middleware, and how the configuration files are used, with the assumption that the Forms servlet is used to generate the initial HTML page. For example, assume the Web server is running on port 8888 on a computer called "example.com". Also assume no modifications have been made to the standard configuration created during the Oracle Fusion Middleware installation process.

When a user runs an Oracle Forms Services application, the following sequence of events occur:

1. The user starts the Web browser and goes to a URL such as:

   ```
   http://example.com:8888/forms/frmservlet?config=myapp&form=hrapp
   ```

   In this example, the top level form module to be run is called "hrapp" using the configuration section called "myapp".

2. Oracle HTTP Server listener receives the request. It finds `/forms` path in the URL and forwards the request to the correct Oracle WebLogic Managed Server based on the WebLogic handler mappings. The mapping is defined in `forms.conf`.

3. Oracle WebLogic Managed Server maps the request to the Oracle Forms Services application that has a context root named `/forms`. It maps the request to the Forms servlet using the frmservlet mapping specified in the `web.xml` file.

4. The Forms servlet running on the Oracle WebLogic Managed Server processes the request. The Forms servlet:
   - Opens the servlet configuration file (`formsweb.cfg` by default), which is located in `$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config`.
   - Determines which configuration section to use in the `formsweb.cfg` file. In this example, the URL contains the query parameter `config=myapp`, therefore, the [`myapp`] section is used.
   - Determines which baseHTML file to use, based on (a) what browser (user-agent) made the request, (b) what platform the browser is running on, and (c) the settings of various parameters in the `formsweb.cfg` file (specifically, `basejpi.htm`, and `base.htm`).
   - Reads the baseHTML file, and returns the contents as an HTML page to the user's Web browser, after performing variable substitutions as follows:

     Whenever a variable (like `%myParam%`) is encountered, the Forms servlet looks for a matching URL query parameter (for example, `&myParam=xxx`), or, failing that, looks for a matching parameter in the `formsweb.cfg` file. If a matching parameter is found, the variable (`%myParam%`) is replaced with the parameter value.

     In this example, the baseHTML file contains the text `%form%`. This is replaced with the value "hrapp".
5. Depending on which baseHTML file the Forms servlet selected, the HTML page returned to the Web browser contains an applet, object or embed tag to start the Forms applet (thin client). The Forms client runs in the JVM environment provided by Sun's Java plug-in.
6. In order to start the Forms applet, its Java code must first be loaded. The location of the applet is specified by the applet codebase and archive parameters.

   The virtual path definition in the `weblogic.xml` file for `/forms/java` allows the applet code to be loaded from the Web server.

   Note: The Forms applet code is only loaded over the network the first time the user runs an Oracle Forms Services application or if a newer version of Oracle Forms Services is installed on the Web server. Otherwise, it is loaded from the cache of the Java plug-in on the local disk.
7. Once the Oracle Forms Services applet is running, it starts a Forms session by contacting the Forms Listener servlet at URL `http://example.com:8888/forms/lservlet`.
8. The Oracle HTTP Server listener receives the request. It forwards the request to Oracle WebLogic Managed Server, since the path `/forms/lservlet` matches a servlet mapping in the `web.xml` file (the one for the Forms Listener servlet).
9. The Forms Listener servlet (lservlet) starts a Forms run-time process (`frmweb.exe` or `frmweb`) for the Forms session.
10. Communication continues between the Forms applet and the Forms run-time process, through the Listener Servlet, until the Forms session ends.
11. The attribute value in a URL (such as the name of the form to run) is passed to the Forms run-time process. Part of the serverArgs value in the baseHTML file is `%form%`, which is replaced by "hrapp". Therefore, the run-time process runs the form in the file "hrapp.fmx".

This file must be present in any of the directories named in the FORMS_PATH environment setting, which is defined in the environment file (default.env by default). You can also specify the directory in formsweb.cfg (for example, form=c:\<path>\myform).

12. The Forms sessions end when either of the following occurs:
    - The top-level form is exited (for example, by the PL/SQL trigger code which calls the "exit_form" built-in function). The user is prompted to save changes if there are unsaved changes. exit_form(no_validate) exits the form without prompting.
    - If the user quits the Web browser, any pending updates are lost.

3.2 Configuration Files

This section introduces the basic files used to configure Forms applications. For more advanced configuration topics, see Chapter 4, "Configuring and Managing Forms Services."

This section contains the following:

- Section 3.2.1, "Oracle Forms Configuration Files"
- Section 3.2.2, "Forms Java EE Application Deployment Descriptors"
- Section 3.2.3, "Oracle HTTP Listener Configuration File"
- Section 3.2.4, "Standard Fonts and Icons File"
- Section 3.2.5, "baseHTML Files"
- Section 3.2.6, "WebUtil Configuration Files"

> **Note:** Location of files are given relative to the DOMAIN_HOME and ORACLE_INSTANCE directory. Forward slashes should be replaced by back slashes on Windows. For more information on terminology used such as Middleware home, Oracle home, Oracle instance, and so on, see the *Oracle Fusion Middleware Administrator's Guide*.

3.2.1 Oracle Forms Configuration Files

Oracle Forms configuration files allow you to specify parameters for your Forms. You can manage these files through the Oracle Enterprise Manager Fusion Middleware Control. These configuration files include:

- default.env
- formsweb.cfg
- ftrace.cfg

> **Note:** For a list of Forms configuration files and their respective locations, refer to Table C–1.

3.2.1.1 default.env

Location: $DOMAIN_HOME/config/fmwconfig/servers/<MANAGED_SERVER>/applications/<appname>_<appversion>/config

Typically, this location is $DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config

This file contains environment settings for Forms run time. On UNIX and Linux, default.env includes the PATH and LD_LIBRARY_PATH.

For a sample default.env file, see Appendix C.3, "Platform Specific default.env Files."

For more information about default.env, see Chapter 4.3, "Managing Environment Variables."

3.2.1.2 formsweb.cfg

Location: $DOMAIN_HOME/config/fmwconfig/servers/<MANAGED_SERVER>/applications/<appname>_<appversion>/config

Typically, this location is $DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config

This Forms configuration file contains the following:

- Values for Forms run-time command line parameters, and the name of the environment file to use (envFile setting).
- Most of the servlet configuration parameter settings that you set during installation. You can modify these parameters, if needed.

Variables (%variablename%) in the base.htm file are replaced with the appropriate parameter values specified in the formsweb.cfg file and from query parameters in the URL request (if any).

For a sample formsweb.cfg file, see Appendix C.2, "Default formsweb.cfg."

For more information about formsweb.cfg, see Chapter 4.2.2, "Configuring Parameters with Fusion Middleware Control."

3.2.1.3 ftrace.cfg

Location: $ORACLE_INSTANCE/config/FormsComponent/forms/server

This file is used to configure Forms Trace. Forms Trace replaces the functionality that was provided with Forms Runtime Diagnostics (FRD) and Performance Event Collection Services (PECS), which were available in earlier releases of Oracle Forms. Forms Trace traces the execution path through a form (for example, steps the user took while using the form).

For more information about ftrace.cfg, see Chapter 12, "Tracing and Diagnostics."

3.2.2 Forms Java EE Application Deployment Descriptors

The Forms Services Java EE application EAR (Enterprise Archive) file formsapp.ear is deployed to the WLS_FORMS (Oracle WebLogic Managed Server) when you configure Oracle Forms.

This results in the creation of a directory structure under $DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string1>/APP-INF directory that is similar to the following:

```
./APP-INF
./APP-INF/lib
./APP-INF/lib/frmconfig.jar
./APP-INF/lib/frmconfigmbeans.jar
./META-INF
```

```
./META-INF/application.xml
./META-INF/jazn-data.xml
./META-INF/jps-config.xml
./META-INF/mbeans.xml
./META-INF/weblogic-application.xml
```

This following directory structure is created under `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string2>/war/WEB-INF` directory.

```
./WEB-INF
./WEB-INF/lib
./WEB-INF/lib/frmsrv.jar
./WEB-INF/web.xml
./WEB-INF/weblogic.xml
```

Note: The sub-directories in `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1` are created by the nostage deployment process of Oracle WebLogic Server. They are named with a random string. For example, e18uoi, wb1h9e and so on.

Deployment descriptors:

- `application.xml` and `weblogic-application.xml` define the structure of the EAR file.
- `web.xml` defines the aliases `frmservlet` and `lservlet` for the Forms servlet and the Forms Listener servlet.
- `weblogic.xml` defines the context parameters and any user defined virtual directory mappings.

For a sample `web.xml` file, see Appendix C.5, "web.xml."

3.2.3 Oracle HTTP Listener Configuration File

This section describes the file used to configure Oracle HTTP Listener for Oracle Forms Services.

Location: `$ORACLE_INSTANCE/config/OHS/<OHS INSTANCE NAME>/moduleconf`

`forms.conf` is the Oracle HTTP listener configuration file for Oracle Forms Services. `forms.conf` defines WebLogic handler mappings for the Managed Server where the Forms Services applications are deployed.

3.2.3.1 About Editing forms.conf

`forms.conf` is an Oracle HTTP Server directives file. In Oracle Fusion Middleware, the `forms.conf` file is included in the Oracle HTTP Server configuration directory at `$ORACLE_INSTANCE/config/OHS/<OHS INSTANCE NAME>/moduleconf`.

If you add any custom Oracle HTTP Server directives to `forms.conf`, you must restart the Oracle HTTP Server node where it resides.

For more information about `forms.conf`, see Appendix C.7, "forms.conf."

3.2.3.2 Configuring OHS on a Separate Host

If you choose to configure Oracle HTTP Server on a separate host, then perform the following tasks:

1. Copy the Forms OHS directives file, `forms.conf.backup` from the tier hosting Forms to the tier hosting OHS and rename it to `forms.conf`.

   Source location (on Forms tier):

   ```
   $ORACLE_
   INSTANCE/config/FormsComponent/forms/server/forms.conf.backup
   ```

   Destination location (on OHS tier):

   ```
   $ORACLE_INSTANCE/config/OHS/<OHS Component
   Instance>/moduleconf/forms.conf
   ```

2. Specify the appropriate managed server cluster or the managed server for the default forms Java EE application context root (`/forms`).

   Example of cluster entry:

   ```
   <Location /forms>
           SetHandler weblogic-handler
           WebLogicCluster    <HOSTNAME>:<WLS_PORT>
           DynamicServerList OFF
   </Location>
   ```

   Example of non-cluster entry:

   ```
   <Location /forms>
           SetHandler weblogic-handler
           WebLogicHost = <HOSTNAME>
           WebLogicPort = <PORT>
   </Location>
   ```

3. Make sure that any directories referenced in user-added directives are accessible on the OHS tier.
4. Restart OHS instance on the OHS tier.

3.2.4 Standard Fonts and Icons File

`Registry.dat` is the file that contains the default font, font mappings, and icon information that Forms Services uses.

Location: `$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/forms/registry/oracle/forms/registry`

For a sample of the default `Registry.dat`, see Appendix C.8, "Registry.dat."

For more information about Registry.dat, see Chapter 4.7, "Deploying Fonts, Icons, and Images Used by Forms Services."

3.2.5 baseHTML Files

Location: `$ORACLE_INSTANCE/config/FormsComponent/forms/server/`

The `base.htm` and `basejpi.htm` are used as templates by the Forms servlet when generating the HTML page used to start an Oracle Forms application.

Oracle recommends that you make configuration changes in the `formsweb.cfg` file using Enterprise Manager and avoid editing these files. To change the baseHTML files, create your own versions and reference them from the `formsweb.cfg` file by changing the appropriate settings.

For a sample baseHTML file, see Appendix C.4, "base.htm and basejpi.htm Files."

3.2.6 WebUtil Configuration Files

This section describes the files used to configure WebUtil at run time. For information about using WebUtil at design time, see the Oracle Forms Developer Online Help. WebUtil configuration files include:

- Default webutil.cfg
- Default webutilbase.htm
- Default webutiljpi.htm

3.2.6.1 Default webutil.cfg

Location: `$ORACLE_INSTANCE/config/FormsComponent/forms/server.`

This file provides all of the configuration settings for WebUtil, including:

- Logging Options
- Installation Options
- File Upload and Download Options
- Server Side Logging Options for logging errors and log messages

For a sample of the `webutil.cfg` file, see Appendix C.10, "Default webutil.cfg."

3.2.6.2 Default webutilbase.htm

Location: `$ORACLE_INSTANCE/config/FormsComponent/forms/server/`

This is the default baseHTML file for running a form on the Web using a generic APPLET tag to include a Forms applet with a certificate registration for WebUtil.

For a sample of the `webutilbase.htm` file, , see Appendix C.11, "Default webutilbase.htm."

3.2.6.3 Default webutiljpi.htm

Location: `$ORACLE_INSTANCE/config/FormsComponent/forms/server/`

This is the default baseHTML file for running a form on the Web using the JDK Java Plugin. For example, this file can be used when running a form on the Web with Firefox on UNIX and a certificate registration for WebUtil.

For a sample of the `webutiljpi.htm` file, , see Appendix C.12, "Default webutiljpi.htm."

3.3 Application Deployment

Once you have created your application in Forms Developer, you are ready for application Web deployment. Oracle Forms Services accesses an application in Oracle Fusion Middleware through a specified URL. The URL then accesses the HTTP Listener, which communicates with the Listener Servlet. The Listener Servlet starts a

Forms run-time process (`frmweb.exe` on Windows or `frmweb` on UNIX and Linux) for each Forms Services session.

For more information about how Forms Services run, see Section 3.1, "Oracle Forms Services in Action."

3.3.1 Deploying Your Application

To deploy a basic form with the default parameters set up by the installer:

1. Create your application in Forms Developer and save it.

   The .fmb file is a design time file that can only be opened in Forms Developer. The .fmx file is the run-time file created when you compile the .fmb and is used for Web deployment.

   For more information about Forms Developer, see the Help menu in Forms Developer.

2. Modify the `formsweb.cfg` file so that Oracle Forms Services can access your application module. You edit this file in the **Web Configuration** page of Fusion Middleware Control. For more information, see Section 4.2, "Configuring Forms Services".

   Table 3–1 shows the configuration of an application called "my_application" with a form module called "`form=hrapp.fmx`":

Table 3–1 Example of Configuration Section Parameter Values

Configuration Section Name	Forms Module Name Value
my_application	hrapp.fmx

   When configured, the Oracle Forms Services module hrapp.fmx is accessible on the Web by entering "`...?config=my_application`" in the browser URL (the name of the **Web Configuration** section in `formsweb.cfg`).

   Note: The name of the configuration section must not include spaces and must contain only alphanumeric characters.

3. Make sure the .fmx file location is specified in the `FORMS_PATH` environment variable.

   For example, in Windows, if your .fmx file is located in `d:\my_files\applications`, in the `FORMS_PATH`, include `d:\my_files\applications`. On Windows, use semi-colons to separate directory locations if specifying multiple locations. On UNIX/Linux, use colons for separators. Specify this information in the **Environment Configuration** page for the environment file.

4. To modify an environment file, select the file in the **Environment Configuration** page of Fusion Middleware Control and add or edit environment variables as needed by your application. For example, you can add the environment variable shown in Table 3–2.

Table 3–2 Example of Environment Variable Values

Environment Variable Name	Environment Variable Value
NLS_LANG	`NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1`

If you specified these environment variables in an environment file, specify this environment file in the respective configuration section of the `formsweb.cfg` in the **Web Configuration** page.

5. Enter the name of your application in the URL as shown:

   ```
   http://example.com:8888/forms/frmservlet?
   ```

   where "example" is the hostname of your computer and "8888" is the port used by your HTTP Listener.

   Once you have created a configuration section, add "config=" and the name of the configuration section. In this example, the URL to access `hrapp.fmx` is:

   ```
   http://example.com:8888/forms/frmservlet?config=my_
   application
   ```

3.3.2 Specifying Parameters

There are two ways to predefine parameter values for your Oracle Forms Services applications. You can define parameters by:

- Editing your application settings in the default section of the **Web Configuration** page of Fusion Middleware Control. The default configuration section displays the default values that are used by Oracle Forms Services.
- Managing (adding, editing, copying, deleting) other system and user parameter values in the named application configuration section (see Section 3.3.3, "Creating Configuration Sections in Fusion Middleware Control"). For example, in the configuration section you create for `myApp`, you can add or change these parameters and their values, as shown in Table 3–3.

Table 3–3 Example Configuration Section: Parameter Values for myApp

Parameter Name	Parameter Value
baseHTML	mybase.htm
baseHTMLjpi	mybasejpi.htm
form	hrapp.fmx
userid	scott/tiger@orcl

Note: Parameters specified in the named configuration section of a **Web Configuration** override the settings in the default section.

Note: System Parameters cannot be overridden in the URL, while user parameters can.

3.3.3 Creating Configuration Sections in Fusion Middleware Control

Under the configuration sections you created in step 2 of Section 3.3.1, "Deploying Your Application", you can specify parameters for your Oracle Forms Services applications. You can specify any application and system parameters that are available in the default section for **Web Configuration** page.

For example, you can set the look and feel of the application to the Oracle look and feel by setting the `lookAndFeel` parameter to the value of `oracle` and clicking **Apply**.

You can also override the default parameter values in the named configuration section. For example, to predefine the connect information of an application to `scott/tiger@orcl`, the parameter value for `userid` must be set in the named configuration section by changing the parameter value of `userid` to `scott/tiger@orcl`.

For other parameters that you can edit, see Chapter 4.2.5, "Forms Configuration Parameters."

3.3.3.1 Editing the URL to Access Oracle Forms Services Applications

You can directly type parameters in the URL that accesses your Oracle Forms Services application. Using the previous example, instead of specifying the `form` parameter in your configuration file, you could also type it into the URL as follows:

```
http://example.com:8888/forms/frmservlet?config=my_application&form=hrapp
```

You can use the ampersand (&) to call a combination of a form and named configuration parameters. In the above example, you are calling the form "hrapp" with the parameter settings you specified in "my_application".

Note: Parameters specified in the URL override the parameters set in the configuration section. See Chapter 4.5, "Managing URL Security for Applications" for more information.

3.3.4 Specifying Special Characters in Values of Runform Parameters

Certain considerations apply if values passed to runform parameters contain special characters. This section describes these considerations, and compares the default behavior in this release with the behavior in prior releases.

Runform parameters are those that are specified in the serverArgs applet parameter of the template HTML file. The value specified for the serverArgs parameter in the template HTML file, after variable substitution, is sometimes referred to as the command-line parameters string. It consists of a series of blank-separated `name=value` pairs. The name must consist solely of alphanumeric or underscore characters. The value portion of a name=value pair can be an arbitrary string.

3.3.4.1 Default Behavior in the Current Release

The value of a runform parameter can be specified in one of three places:

1. In the value of the `serverArgs` parameter in the template HTML file (for example, base.htm).
2. In the value of a variable specified in the configuration file (for example, `formsweb.cfg`), which is substituted (directly or recursively) for a variable reference in (1). Such values are typically maintained using Fusion Middleware Control; see Chapter 4.2, "Configuring Forms Services."

3. As an attribute value in a URL, which is substituted directly for a variable reference in (1) or (2).

For case (3), URL syntax rules (as enforced by the browser and the application server) require that certain characters be entered as URL escape sequences ('%' followed by 2 hexadecimal digits representing the ASCII value of the character, for a total of three characters).

This requirement includes the `%` character itself (which must be entered as `%25`). In addition, Oracle Forms Services currently requires that the quote character ('"') be entered as `%22`, even if the browser and the application server allow a quote to be entered without escaping.

URL syntax rules also allow a space to be entered as a + (as an alternative to the URL escape sequence `%20`). However in the value of the `otherparams` configuration parameter, a + is treated specially; it separates name=value pairs as opposed to indicating a space embedded in the value of a runform parameter.

For example, if a runform application has user parameters `param1` and `param2`, and you want to assign them the values 'a b' and 'c d', you do so by incorporating the following into a URL:

```
&otherparams=param1=a%20b+param2=c%20d
```

When specifying runform parameters in the template HTML files or in the configuration files (cases (1) and (2)), Forms requires URL escape sequences in some circumstances, allows them in others, and forbids them in still others.

Outside of the values of runform parameters, URL escape sequences must not be used. For example, the = in a name=value pair must always be specified simply as =, and the space that separates two adjacent name=value pairs must always be specified simply as " " (a single space character).

Within the value of a runform parameter, space (' ') must be specified as a URL escape sequence (`%20`). The HTML delimiter character (specified in the configuration file) must also be specified as a URL escape sequence. And when the runform parameter is specified in the template HTML file (case (1)), quote ('"') must also be specified as a URL escape sequence (%22).

Any other 7-bit ASCII character may also be specified as a URL escape sequence, although this is not required (except possibly for `%`, as noted below). Certain additional restrictions apply to the `%` character. These include:

- If the HTML delimiter is `%` (the default), then an occurrence of `%` within the value of a runform parameter must be escaped (specified as `%25`). (This actually follows from the requirement stated above, that the HTML delimiter character be escaped). Furthermore, variable names must never begin with two hexadecimal digits that represent a 7-bit ASCII value (that is, two hexadecimal digits, the first of which is in the range 0-7).
- If the HTML delimiter is not `%`, then an occurrence of `%` must be escaped if it is immediately followed by an octal digit and then a hexadecimal digit. It is recommended that other occurrences of '%' also be escaped; but this is not a requirement.

(You might choose to ignore this recommendation if you have existing template HTML files or configuration files created in prior releases, which use an HTML delimiter other than '%', and which contain '%' in runform parameter values).

3.3.4.2 Behavior in Previous Releases

Release 9.0.4 and later behave the same as the current release except that a quote must be escaped (%22) within the value of a runform parameter in a configuration file, and in the template HTML file.

Releases before 9.0.4 did not allow URL escape sequences in runform parameter values specified in the template HTML file or the configuration file (cases (1) and (2) above). In all three cases, it was difficult or impossible to specify certain special characters, notably space, quote, and apostrophe. Also, certain transformations were applied to the parameter value before passing it to runform. Most notably, if a value began and ended with an apostrophe, these were typically stripped off. However, these transformations were not well-defined, and they differed between the Web and client/server environments.

3.3.4.3 Obtaining the Behavior of Prior Releases in the Current Release

If your applications are dependent on the behavior of prior releases, you can obtain that behavior in the current release, by simply setting the value of the `escapeparams` variable to `False` in the configuration file (this can be accomplished using Fusion Middleware Control).

If you want to obtain the old behavior only for selected applications, you can specify different values for the `escapeparams` variable in different configuration sections. Applications that require the old behavior can specify a configuration section in which the `escapeparams` variable is set to `False`; applications that require (or tolerate) the behavior in the current release can specify a configuration section in which the `escapeparams` variable is set to `True`.

3.3.4.4 Considerations for Template HTML Files

If you are creating your own template HTML files, then bear in mind the following:

It is recommended that a reference to the `escapeparams` variable (the string `%escapeparams%`, if '`%`' is the HTML delimiter character) appear at the beginning of the value of the `serverArgs` applet parameter, followed by a space. See the shipped base.htm file for an example.

References to the `escapeparams` variable must appear nowhere else in the template HTML file. If you choose to enclose the value of the `serverArgs` applet parameter in apostrophes instead of quotes, then within the value of a runform parameter in your template HTML file, apostrophes must be escaped (%27). Quotes do not require escape sequences.

It is permissible to omit the reference to the `escapeparams` variable from the beginning of the value of the `serverArgs` applet parameter. This results in the behavior of prior releases, regardless of the value specified in the configuration file for the `escapeparams` variable.

3.3.4.5 Considerations for Static HTML Pages

If you are invoking the runform engine using static HTML, and you want to obtain the behavior in the current release, then you must take certain steps.

The basic rule is that your static HTML must look like the HTML generated by the Forms servlet. Specifically, the value of the `serverArgs` applet parameter must begin with the string `escapeparams=true` (case-insensitive).

Also, in the value portion of each name=value pair, in the value of the `serverArgs` applet parameter, certain characters must be specified by a URL escape sequence, as listed in Table 3–4:

Table 3–4 URL Escape Sequences for Static HTML pages

Characters that must be escaped	URL Escape Sequence
newline ' \n '	%0a
space ' '	%20
quote ' " '	%22
percent ' % '	%25
apostrophe ' ' '	%27
left parenthesis ' ('	%28
right parenthesis ') '	%29

It is also permissible to escape other 7-bit ASCII characters in the value portion of a `name=value` pair.

Here's an example of what the `serverArgs` applet parameter might look like in static HTML. This is for a form named "my form" (quotes not included), which is being passed the value "foo'bar" (quotes again not included) to the user-defined parameter named `myparam`.

```
<PARAM NAME="serverArgs" VALUE="escapeparams=true module=my%20form
userid=scott/tiger@mydb myparam=foo%27bar">
```

3.3.5 Accessing the Listener Servlet Administration Page

You can display a test page for the Listener Servlet by accessing the following URL:

`http://<hostname>:<port>/forms/frmservlet/admin`

The information displayed depends on the value of the initialization parameter `TestMode`. This parameter is set in the `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF/web.xml` file. An example is shown below:

```
<init-param>
<!-- Display sensitive options on the /admin page ? -->
    <param-name>TestMode</param-name>
    <param-value>true</param-value>
</init-param>
```

3.4 Client Browser Support

Users can view Oracle Forms applications on the Web using Sun's Java Plug-in. In future patch releases other virtual machines may be supported.

For more information about client browser support, including the latest supported platforms, go to the Forms Developer menu and choose **Help | Forms on OTN...** to locate the Client Platform Statement of Direction.

You can also find information on certification on OTN at http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

3.4.1 How Configuration Parameters and BaseHTML Files are Tied to Client Browsers

When a user starts a Web-enabled application (by clicking a link to the application's URL), the Forms servlet:

1. Detects which browser is being used.
2. Selects the appropriate baseHTML file using Table 3–5:

Table 3–5 baseHTML file descriptions

Detected Browser	Base HTML file used
Internet Explorer	`basejpi.htm`
Mozilla FireFox 3.0	`basejpi.htm`
All other browsers and Macintosh clients	`base.htm`

3. Replaces variables (%*variablename*%) in the baseHTML file with the appropriate parameter values specified in the Forms `servlet.initArgs` file, `formsweb.cfg` file, and from query parameters in the URL request (if any).
4. Sends the HTML file to the user's browser.

3.4.2 Forms Single Sign-On on Mozilla 3.x

Ensure that you have enabled cookies from Oracle site when using Forms and Single Sign-On on Mozilla 3.x. To enable the cookies, perform the following steps:

1. Open Mozilla Firefox, select Tools.
2. Select Options and then Privacy.
3. Select the Accept Cookies from site box.

   These steps are not required for other browsers.

   For more information on default browser settings in Mozilla, refer to `http://www.mozilla.com`.

4

Configuring and Managing Forms Services

This chapter contains the following sections:

4.1 Fusion Middleware Control and Oracle Forms

The Fusion Middleware Control is a Web-based tool that you launch from your default browser. The default URL is:

```
http://<example.com>:7001/em
```

Use the Web-based Oracle Enterprise Manager Fusion Middleware Control to:

- Monitor metrics for a Forms Services instance. See Section 14.1.1.1, "Monitoring Forms Services Instances" for more information.
- Manage user sessions. See Section 4.4, "Managing User Sessions" for more information.
- Configure parameters for a Forms Services instance. See Section 4.2.2, "Configuring Parameters with Fusion Middleware Control" for more information.
- Configure Forms Trace and monitor trace metrics. See Section 12.2, "Enabling and Configuring Forms Trace" and Section 12.6, "Taking Advantage of Oracle Diagnostics and Logging Tools" for more information.
- Configure multiple environment files. See Section 4.3, "Managing Environment Variables" for more information.
- Configure and use JVM pooling. See Section 10.8, "Managing JVM Pooling from Fusion Middleware Control" for more information.

4.1.1 Accessing Forms Services with Fusion Middleware Control

To perform most management tasks for a Forms instance using Fusion Middleware Control, you start by navigating to the Forms home page in Fusion Middleware Control.

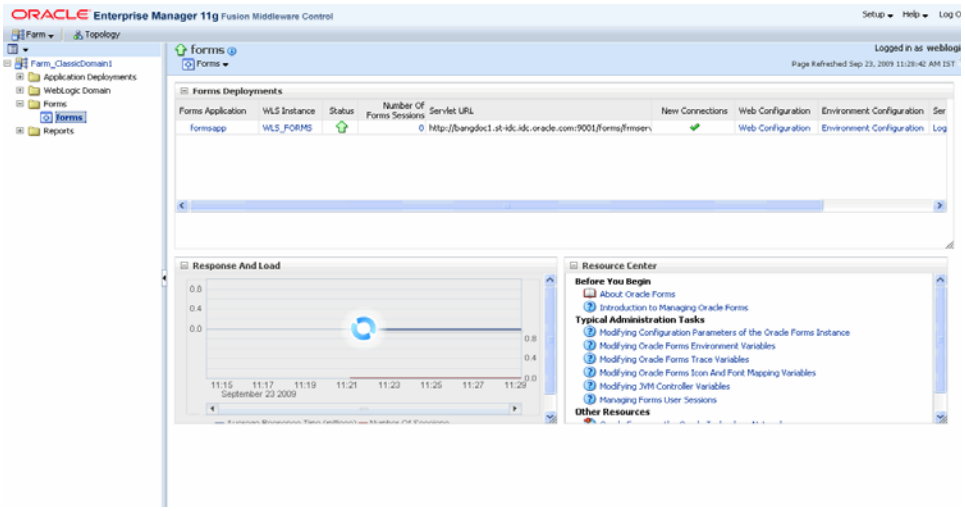
To navigate to the Forms Home page in Fusion Middleware Control:

1. Navigate to the home page for the Fusion Middleware Control that contains the Forms instance you want to manage.

   For introductory information about using the Enterprise Manager Fusion Middleware Control, see "Overview of Oracle Fusion Middleware Administration Tools" in the *Oracle Fusion Middleware Administrator's Guide*.

2. In the Farm pane, click the Fusion Middleware folder, then click the link for the Forms instance. This displays the Forms Home page (Figure 4–1) in the Fusion Middleware Control.

Figure 4–1 Forms Home page



3. The Forms Home page provides information on the Forms applications that are deployed on the Oracle instance. Table 4–1 describes the information displayed on the Forms Home page.

Table 4–1 Forms Deployment Fields

Field	Description
Forms Application	Lists the names of the Forms applications that are deployed on the Oracle WebLogic Server instance. Click the name to view the Forms application home page.
WLS Instance	Name of Oracle WebLogic Server instance where the application is deployed.
Status	Indicates the status of the forms application. A green up arrow indicates the application is running. A red down arrow indicates the application is not started.
Number of Forms Sessions	Displays the number of active forms sessions.
Servlet URL	Displays the URL for the Forms servlet.
New Connections	Indicates whether new connections are enabled or not.

Table 4–1 (Cont.) Forms Deployment Fields

Field	Description
Web Configuration	Link to the Web Configuration page.
Environment Configuration	Link to the Environment Configuration page.
Servlet Logs	Link to the Servlet Logs.

To access the Forms Menu in Fusion Middleware Control:

1. Navigate to the Forms home page in Fusion Middleware Control.
2. Click **Forms** on the top left. This displays the Forms Menu. Table 4–2 lists the Menu Selections that are available in the Forms Menu.

Table 4–2 Forms Menu Options

Select	To Display
Home	Forms Home page. This page displays a list of the Forms deployments and their details. This page also displays the Response and Load statistics and a set of useful links in the Resource Center.
Monitoring - Performance Summary	Performance Summary page. This page displays a set of default performance charts that show the values of specific performance metrics. For more information, see the *Oracle Fusion Middleware Performance Guide*.
Monitoring - Servlet Log	Log Messages page. Oracle Fusion Middleware components generate log files containing messages that record all types of events.
JVM Controllers	JVM Controllers page. This page is used to manage the JVM controller for the Forms instance.
User Sessions	User Sessions page. This page is used to monitor and trace User Sessions within a Forms instance.
Web Configuration	Web Configuration page. This page is used to configure deployment of Forms applications and manage configuration sections and parameters in `formsweb.cfg`.
Trace Configuration	Trace Configuration page. This page is used to manage the settings used for tracing of user sessions.
Fonts and Icons Mapping	Fonts and Icons Mapping page. This page is used to change, add, or delete parameters in the Registry.dat file.
JVM Configuration	JVM Configuration page. This page is used to modify the JVM controllers that can be subsequently spawned for the Forms instance.
Environment Configuration	Environment Configuration page. This page is used to manage environment variables that define environment settings for Forms run time.
Associate/Disassociate OID	Associate/Disassociate OID page. This page is used to associate and disassociate a forms deployment with an Oracle Internet Directory host to enable Single Sign-On functionality.
General Information	Displays information about the Target Name, Version, Oracle Home, Oracle Instance, and Host.

Note: For the pages that include a **Help** icon, click the **Help** icon to access the page-level help. The page-level help describes each element in the page.

4.2 Configuring Forms Services

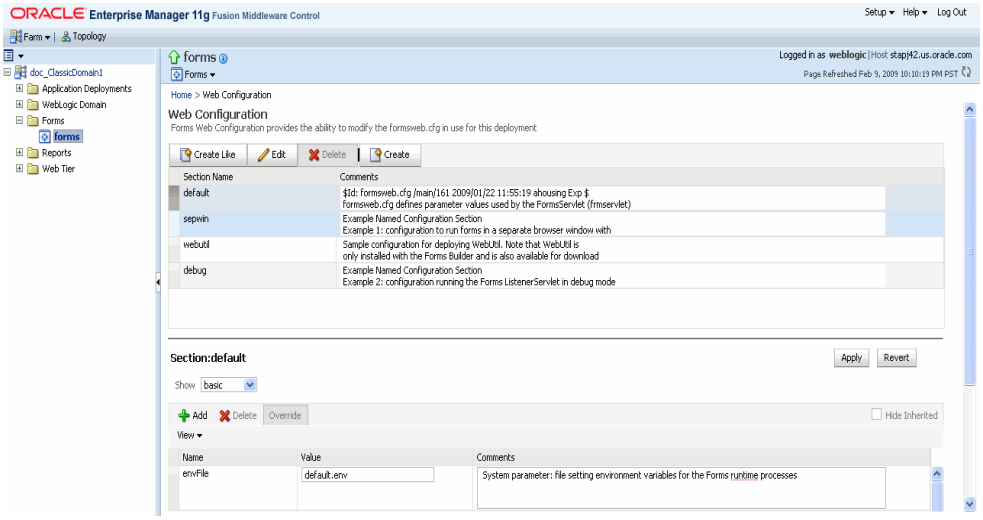
Use the **Web Configuration** page in Fusion Middleware Control to configure deployment of Forms applications by modifying `formsweb.cfg`.

To access Web Configuration page:

1. Start Fusion Middleware Control.
2. From the Fusion Middleware Control main page, click the link to the Oracle Forms Services instance that you want to configure.
3. From the Forms menu list, select **Web Configuration**.

   The **Web Configuration** page (Figure 4–2) is displayed.

Figure 4–2 Web Configuration Page



4. See Table 4–3 and Table 4–4 for the tasks that you can do.

Note: As with most Web applications, it is easy to lose unsaved changes by switching pages. Be sure to save any changes you make through Fusion Middleware Control to Forms configuration or environment files before proceeding to other pages.

The length of time it takes for changes to be saved is affected by the number of lines you have changed. For example, an additional fifty lines of comments takes longer to save than just the deletion of a single entry.

4.2.1 Common Tasks in the Web Configuration Page

Table 4–3 describes the common tasks that you can do to edit configuration with the sections of a configuration file and their parameters.

Table 4–3 Common Tasks for Working with Configuration Sections

Task	Description	Comment
Create Like	Creates a copy of a configuration section.	Use to create a configuration section based on the parameters of an existing configuration section.
Edit	Opens the Edit Description dialog.	Allows editing of the text description of a configuration section.
Delete	Opens the Confirmation dialog when deleting a configuration section.	Irrevocably deletes a configuration section and its contents when you click Delete in the Confirmation dialog.
Create	Opens the Create Section dialog.	Creates a configuration section. You must supply a required name and an optional description for it.

Table 4–4 describes the tasks that you can do to modify the parameters within a named configuration section:

Table 4–4 Common Tasks for Working with Parameters

Task	Description	Comment
Show	Drop down list for selecting named groups of parameters in a configuration section.	Use for viewing and editing groups of parameters. The groups of parameters include: ■ basic ■ sso ■ trace ■ plugin ■ HTML ■ applet ■ advanced ■ all For more information, see Section 4.2.5, "Forms Configuration Parameters".
Revert	Enables you to revert all changes made to parameters in a configuration section since the last apply.	Does not allow you to revert individual changes in a configuration section.
Apply	Applies and activates all changes made to parameters in a configuration section.	Once applied, you cannot revert changes to individual parameters.
Hide Inherited	Enables you to hide or display parameters that are inherited from a parent configuration section.	Use this to view parameters that have been explicitly added to a configuration section or to view all parameters (including those that are inherited from the default section).
Add	Displays the Add Parameter dialog.	Add a parameter to a configuration section based on a mandatory name and an optional value and description.

Table 4–4 (Cont.) Common Tasks for Working with Parameters

Task	Description	Comment
Delete	Deletes a parameter.	There is no Confirmation dialog. Once applied, you cannot revert changes to individual parameters.
Override	Allows overriding and editing of a parameter which is inherited from the default section.	Click Apply to save and activate your changes.

4.2.2 Configuring Parameters with Fusion Middleware Control

For a description and the location of the Forms servlet configuration file (`formsweb.cfg`), see Section 3.2.1.2, "formsweb.cfg".

4.2.2.1 Parameters that Specify Files

Three configuration parameters specify files. Of these, two baseHTML parameters must point to appropriate `.htm` files. Typically, the following values and their parameters should appear in the default configuration section, as shown in Table 4–5.

Table 4–5 Default Configuration Parameters that Specify Files

Parameter	Value
baseHTML	`base.htm`
baseHTMLjpi	`basejpi.htm`
envFile	`default.env`

All of these parameters specify file names. If no paths are given (as in this example), the files are assumed to be in the same directory as the Forms servlet configuration file (`formsweb.cfg`), that is `$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config`

4.2.3 Managing Configuration Sections

This section describes creating, editing, duplicating, and deleting named configuration sections.

4.2.3.1 Creating a Configuration Section

You can create a configuration section in `formsweb.cfg` from the **Web Configuration** page of Fusion Middleware Control. These configurations can be requested in the end-user's query string of the URL that is used to run a form.

To create a configuration section:

1. Start the Enterprise Manager Fusion Middleware Control.
2. From the Fusion Middleware Control main page, click the link to the Forms Services instance that you want to configure.
3. From the Forms menu list, select the **Web Configuration**.
4. Click **Create** at the top of the **Web Configuration** region.

   The **Create Section** dialog appears.
5. Enter a name and description for the configuration section and click **Create**.

Note: The name must not contain any special characters such as #, *.

The configuration section is added.

For example, to create a configuration to run Forms in a separate browser window with the `Oracle` look and feel, create a section called sepwin and add the following parameters from Table 4–6:

Table 4–6 Sample Parameters to Add to a Configuration Section

Parameter	Value
form	`<module>`
separateFrame	`True`
lookandfeel	`Oracle`

Your users would type the following URL to launch a form that uses the "sepwin" (or the name you applied) configuration:

```
http://server:port/forms/frmservlet?config=sepwin
```

4.2.3.2 Editing a Named Configuration Description

You can edit the description (comments) for a named configuration from the **Web Configuration** page.

Note: You can make a backup of the configuration section you are about to edit by duplicating it first. For more information, see Section 4.2.3.3, "Duplicating a Named Configuration"

To edit a named configuration description:

1. In the **Web Configuration** region, select the row containing the configuration section you want to edit.
2. Click **Edit**.
3. The **Edit Description** dialog appears.
4. Enter the text for the comment.
5. Click **Save**.

   The **Edit Description** dialog box is dismissed, and your changes are saved.

4.2.3.3 Duplicating a Named Configuration

You can make a copy of a named configuration for backup purposes, or create configuration sections from existing configurations or other duplicates.

To duplicate a named configuration:

1. In the **Web Configuration** region, select **Create Like**.
2. In the Create Like dialog, from the **Section to Duplicate** menu list, select the name of an existing configuration section you want to duplicate.
3. In the **New Section Name** field, enter a name for the configuration section. The name for the configuration section must be unique.

4. Click **Create**.

   A section with the same parameters, parameter values and comments of the section you are duplicating is created.

4.2.3.4 Deleting a Named Configuration

When you delete a named configuration, you delete *all* the information within it. If you only want to delete specific parameters, see Section 4.2.4, "Managing Parameters".

To delete a named configuration:

1. From the **Web Configuration** region, select the row of the configuration section you want to delete.
2. Click **Delete**.

   The **Confirmation** dialog appears.
3. Click **Delete**.

   The configuration section is deleted.

   Oracle Enterprise Manager returns to the **Web Configuration** page and displays the remaining configurations.

> **Note:** You cannot delete the Default configuration section.

4.2.4 Managing Parameters

Use Fusion Middleware Control to manage parameters within a named configuration. You can add, edit, or delete parameters from the Section pane of Fusion Middleware Control.

To edit a new or overridden parameter in a configuration section:

1. From the **Web Configuration** region, select the row of the configuration section that contains the parameter(s) you want to edit.
2. In the Section region, select the parameter group from the **Show** menu list. The parameters of the group are displayed.
3. Select the row of the parameter you want to edit. Enter the Value and Comments.

> **Note:** You can edit new or overridden parameters. Inherited parameters must first be overridden so they can be edited. In Figure 4–3, `test1` is an example of a new parameter and `lookandfeel` is an example of an overridden parameter.

4. Click **Apply** to save the changes or **Revert** to discard them.

To add a parameter to a configuration:

1. In Fusion Middleware Control, from the **Web Configuration** region, select the configuration section row to which you want to add a parameter.
2. Click Add to add a parameter.

   The Add dialog box is displayed.
3. Enter the Name, Value and Comments for the parameter.

4. Click **Create** to add the parameter.
5. Click **Apply** to save the changes or **Revert** to discard them.

To delete a parameter in a configuration:

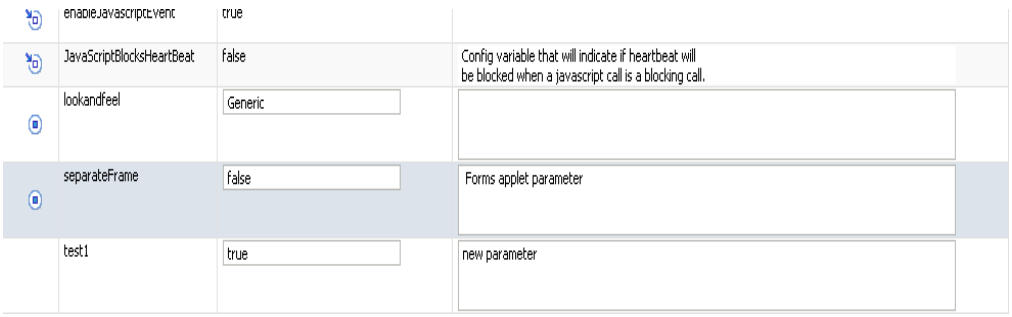
1. In Fusion Middleware Control, from the **Web Configuration** region, select the configuration section row that contains the parameter you want to delete.
2. In the Sections region, from the Show menu list, select the parameter group that contains the parameter you want to delete.
3. Select the row that contains the parameter you want to delete.
4. Click **Delete**.
5. Click **Apply** to save the changes or **Revert** to discard them.

Note: You can delete/edit multiple parameters at a time.

Note: You can only delete user-defined parameters. Inherited parameters (such as `enableJavascriptEvent` in Figure 4–3) cannot be deleted.

Note: When you delete an overridden parameter, the parameter is not deleted but instead regains its inherited status.

Figure 4–3 Parameter States



4.2.5 Forms Configuration Parameters

The section provide information about Forms configuration parameters. These parameters can be specified in the Forms configuration file (`formsweb.cfg`), as described in preceding sections. Many of these parameters can also be specified in the URL. Parameters that cannot be specified in the URL are listed in Section 4.2.5.8. A value in the URL overrides a value from `formsweb.cfg`. The following notes apply to all the parameter tables from Section 4.2.5.1 to Section 4.2.5.7:

- Required/Optional: A parameter is required if the Forms Services requires a non-null value (from `formsweb.cfg` or, where allowed, from the URL) to function correctly.
- Default values: For required parameters, the parameter description lists the default value from the default section of the `formsweb.cfg` that is shipped with the Forms product (or at least indicates that it specifies an appropriate value).

For optional parameters, the parameter description may show a non-null default value from the default section of the `formsweb.cfg` that is shipped with the Forms product. In addition, the parameter description may show the default value that is assumed if no value is specified. (This is the non-null value that produces the same behavior as a null value). When the description for an optional parameter simply shows an unqualified default value, the implication is that this value is both the default value from the default section of the `formsweb.cfg` that is shipped with the Forms product, and also the default value that is assumed if no value is specified.

When the description for an optional parameter does not explicitly specify a default value, the implication is that the default value is null.

- Runform parameters: The descriptions for some parameters indicate that they are runform parameters. They are passed to the frmweb process using the serverArgs applet parameter. For such a parameter, the syntax rules documented in Section 3.3.4 must be adhered to when specifying a value that contains special characters.
- Sub-arguments for otherparams: The descriptions for some parameters indicate that they are sub-arguments for otherparams. That means that in order for the parameter to take effect (when specified in `formsweb.cfg` or the URL), it must appear in the form "`name=%name%`" within the value of the otherparams parameter. So, for example, if you are adding the parameter "array" (with a value of "no") to a configuration section, you must also add "`array=%array%`" to the value of the otherparams parameter.

  Note that these parameters are all runform parameters (since the otherparams parameter is itself a runform parameter), and so the syntax rules documented in Section 3.3.4 must be adhered to when specifying a value that contains special characters.

This section includes:

4.2.5.1 Basic Configuration Parameters

These basic parameters control the behavior of the Forms servlet. These parameters are described in Table 4–7:

Table 4–7 Basic Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
envFile	Required	Specifies the name of the environment configuration file. Default value from formsweb.cfg is default.env.
form	Required	Specifies the name of the top level Forms module (fmx file) to run. Default value from formsweb.cfg is test.fmx. This parameter is a runform parameter.
height	Required	Specifies the height of the form applet, in pixels. Default value from formsweb.cfg is 600.
userid	Optional	Login string. For example: scott/tiger@ORADB. This parameter is a runform parameter.
width	Required	Specifies the width of the form applet, in pixels. Default value from formsweb.cfg is 750.

4.2.5.2 Single Sign-On Configuration Parameters

Table 4–8 Single Sign-On Configuration Parameters

Parameter	Required / Optional	Parameter Value and Description
ssoCancelUrl	Optional	Specifies the Cancel URL for the dynamic resource creation DAS page.
ssoDynamicResourceCreate	Optional	Specifies whether dynamic resource creation is enabled if the resource is not yet created in the OID. Default value is true.
ssoErrorUrl	Optional	Specifies the URL to redirect to if ssoDynamicResourceCreate is set to false.
ssoMode	Optional	Specifies whether the URL is protected in which case, mod_osso is given control for authentication or continue in the FormsServlet if not. Set it to true in an application-specific section to enable Single Sign-On for that application. Default value is false.
ssoProxyConnect	Optional	Specifies whether session should operate in proxy user support or not. Set ssoProxyConnect to yes to enable for particular application. Default value is no. This parameter is a sub-argument for otherparams.

4.2.5.3 Trace Configuration Parameters

Table 4–9 Trace Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
`debug`	Optional	Allows running in debug mode. Default value is `No`. This parameter is a runform parameter.
`EndUserMonitoringEnabled`	Optional	Indicates whether End User Monitoring integration is enabled. Default value is `false`.
`EndUserMonitoringURL`	Optional	Indicates where to record End User Monitoring data.
`host`	Optional	Specifies the host for the debugging session. This parameter should be used for debugging purposes only. It identifies the host on which the forms engine process is started. This parameter is a runform parameter.
`log`	Optional	Supports tracing and logging. The value of this parameter, if set, is the file name of the trace log file. This parameter is a sub-argument for otherparams.
`port`	Optional	Port to use for debugging. This parameter should be used for debugging purposes only. The value of this parameter identifies the port on which the forms engine process is listening. If not specified, the default value is 9000. This parameter is ignored if serverURL has been specified. This parameter is a runform parameter.
`record`	Optional	Supports tracing and logging. This parameter is a sub-argument for otherparams.
`tracegroup`	Optional	Supports tracing and logging. This parameter is a sub-argument for otherparams.

4.2.5.4 Plug-in Configuration Parameters

These parameters are for use with Sun Java Plug-in.

Table 4–10 Sun Java Plug-in Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
archive	Optional	Comma-delimited list of archive files that are used or downloaded to the client. For each file, include the file name if the file is in the codebase directory, or include the virtual path and file name. Default value for formsweb.cfg is frmall.jar.
codebase	Required	Virtual directory you define to point to the physical directory ORACLE_HOME/forms/java, where, by default, the applet JAR files are downloaded from. Default value from formsweb.cfg is /forms/java.
imageBase	Optional	Indicates where icon files are stored. Legal values: ■ codeBase, which indicates that the icon search path is relative to the directory that contains the Java classes. Use this value if you store your icons in a JAR file (recommended). ■ documentBase, which is the URL pointing to the HTML file. Default value from formsweb.cfg is codeBase. If no value is specified, then the value of documentBase is used.
jpi_classid	Required	Sun's Java Plug-in class ID. formsweb.cfg specifies an appropriate value.
jpi_codebase	Required	Sun's Java Plug-in codebase setting. formsweb.cfg specifies an appropriate value.
jpi_download_page	Required	Sun's Java Plug-in download page. formsweb.cfg specifies an appropriate value.
jpi_mimetype	Required	Parameter related to version of Java Plug-in. formsweb.cfg specifies an appropriate value.

4.2.5.5 HTML Page Configuration Parameters

Table 4–11 HTML Page Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
baseHTML	Required	The default base HTML file. Default value from formsweb.cfg is base.htm.

Table 4–11 (Cont.) HTML Page Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
baseHTMLjpi	Required	Physical path to HTML file that contains Java Plug-in tags. Used as the baseHTML file if the client browser is not on Windows and the client browser is either Firefox or IE without the IE native settings. Default value from formsweb.cfg is basejpi.htm.
HTMLafterForm	Optional	HTML content to add to the page below the area where the Forms application is displayed.
HTMLbeforeForm	Optional	HTML content to add to the page above the area where the Forms application is displayed.
HTMLbodyAttrs	Optional	Attributes for the <BODY> tag of the HTML page.
pageTitle	Optional	HTML page title, attributes for the BODY tag, and HTML to add before and after the form. Default value fromformsweb.cfg is Oracle Fusion Middleware Forms Services.

4.2.5.6 Applet Configuration Parameters

These parameters are specified in the baseHTML file as values for object or applet parameters. They describe the visual behavior and appearance of the applet.

Table 4–12 Applet or Object Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
background	Optional	Specifies the .GIF file that should appear in the background. Set to NO for no background. Leave empty to use the default background.
colorScheme	Optional	Determines the application's color scheme. Legal values: Teal, Titanium, Red, Khaki, Blue, BLAF, SWAN, Olive, or Purple. Default value from formsweb.cfg is teal. **Note**: colorScheme is ignored if LookAndFeel is set to Generic.
logo	Optional	Specifies the .GIF file that should appear at the Forms menu bar. Set to NO for no logo. Leave empty to use the default Oracle logo.
lookAndFeel	Optional	Determines the applications look-and-feel. Legal values: Oracle or Generic (Windows look-and-feel). Default value from formsweb.cfg is Oracle.
separateFrame	Optional	Determines whether the applet appears within a separate window. Legal values: true or false (default).

Table 4–12 (Cont.) Applet or Object Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
splashScreen	Optional	Specifies the .GIF file that should appear before the applet appears. Set to NO for no splash. Leave empty to use the default splash image. To set the parameter include the file name (for example, myfile.gif) or the virtual path and file name (for example, images/myfile.gif).

4.2.5.7 Advanced Configuration Parameters

Table 4–13 Advanced Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
allowAlertClipboard	Optional	Forms applet parameter. Default value is true.
allowNewConnections	Optional	Determines whether new Forms sessions are allowed. This is also used by the Forms Home page in Fusion Middleware Control to show the current Forms status. Default value is true.
applet_name	Optional	Configuration for JavaScript integration. This is name of the Forms applet that can be used to refer to it from a JavaScript code.
array	Optional	Set this parameter to no to suppress array processing. This causes Forms to send only a single row at a time to the database for an INSERT, UPDATE, or DELETE, and it causes the database to return only a single row of query results at a time. This usually results in the first retrieved record displaying faster, but the total time to display all rows in the query result is longer. Default value if not specified is yes. This parameter is a sub-argument for otherparams.
buffer_records	Optional	Set this parameter to yes to set the number of records buffered in memory to the number of rows displayed, plus 3 (for each block). This saves Forms Runtime memory, but may slow down processing because of increased disk I/O. Sub argument for otherparams. Default value if not specified is no. This parameter is a sub-argument for otherparams.
clientDPI	Optional	Specifies the dots per inch (DPI) and overrides the DPI setting returned by the JVM, allowing you to manage varying DPI settings per platform. Oracle recommends that you use an integer between 50 and 200.
connectionDisallowedURL	Optional	This is the URL shown in the HTML page that is not allowed to start a session.

Table 4–13 (Cont.) Advanced Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
`cursorBlinkRate`	Optional	To modify the cursor blink rate, or disable blinking, set the client parameter `cursorBlinkRate` as follows: `<PARAM NAME="cursorBlinkRate" VALUE="1000">`. The default is 600 milliseconds: the cursor completes one full blink every 1.2 seconds (1200 ms). A value of zero disables the blinking and the cursor remains visible all the time.
`debug_messages`	Optional	Set this parameter to `yes` to cause Forms to display ongoing messages about trigger execution while the form runs. Default value if not specified is `no`. This parameter is a sub-argument for otherparams.
`defaultcharset`	Optional	Specifies the character set to be used in servlet requests and responses. Defaults to ISO-8859-1 (also known as Latin-1). Ignored if the servlet request specifies a character set (for example, in the content-type header of a POST). The values of this parameter may be specified either as an IANA character set name (for example, SHIFT_JIS) or as an Oracle character set name (for example, JA16SJIS). It should match the character set specified in the NLS_LANG environment variable, and it should also be a character set that the browser can display. Also, if the browser allows multibyte characters to be entered directly into a URL, for example, using the IME, as opposed to URL escape sequences, and to allow end users to do this, then the value of this parameter should match the character set that the browser uses to convert the entered characters into byte sequences. **Note**: If your configuration file contains configuration sections with names that contain characters other than 7-bit ASCII characters, then the following rules apply. If a config parameter is specified in a URL or in the body of a POST request with no specified character set, and the value contains non-7-bit ASCII characters, then the value is interpreted using a character set named in the defaultcharset parameter. However, only the language-dependent default section and the language-independent default section of the configuration file is searched for the defaultcharset parameter. No other configuration section is searched because the name is not yet known.
`digitSubstitution`	Optional	Determines the BIDI digitSubstitution. Permissible values are `none`, `national`, and `context`. Default value is context.

Table 4–13 (Cont.) Advanced Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
disableMDIScrollbars	Optional	Set this parameter to true to disable horizontal and vertical scrollbars in the Forms main applet window. You can also add this parameter in basejpi.html, in the OBJECT tag: <PARAM NAME="disableMDIscrollbars" VALUE="%disableMDIScrollbars%">. In the tag <EMBED SRC> add disableMDIScrollbars="%disableMDIScrollbars%". Default value if not specified is false.
disableValidateClipboard	Optional	Forms applet parameter. Default value is false.
enableJavascriptEvent	Optional	Configuration for JavaScript integration. Default value is true.
escapeparams	Optional	Set this parameter to false for runform to treat special characters in runform parameters as it did in releases before 9.0.4. This parameter is a Forms run-time argument and specifies whether to escape certain special characters in values extracted from the URL for other run-time arguments. Default value is false.
formsMessageListener	Optional	Forms applet parameter that specifies the class that the Forms client uses to enable recording of Forms messages for Tool Vendor Interface (TVI) / Intercept Server.
heartBeat	Optional	Use this parameter to set the frequency at which a client sends a packet to the server to indicate that it is still running. Define this integer value in minutes or in fractions of minutes, for example, 0.5 for 30 seconds. Default value, if not specified, is 2 minutes. If the heartBeat is less than FORMS_TIMEOUT, the user's session is kept active, even if they are not actively using the form. **Note**: If heartBeat is higher than the parameter session-timeout, then the value of session-timeout takes precedence over heartBeat. To increase the value of heartBeat, the value of session-timeout must be greater than heartBeat. For more information on this parameter, see "Session-timeout" in *Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.
highContrast	Optional	When highContrast is set to true, frame labels are black if foreground and background colors are not specified. Default value is false.

Table 4–13 (Cont.) Advanced Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
`HTMLdelimiter`	Optional	This parameter defines the delimiter for parameters in the base HTML files. Default delimiter is %.
`JavaScriptBlocksHeartBeat`	Optional	Configuration variable that indicates if `HeartBeat` is blocked when a JavaScript call is a blocking call. Default value is `false`.
`legacy_lifecycle`	Optional	Applet parameter for Sun's Java Plug-in. A value of true causes a running applet to be reused when requested. This parameter also affects the contents of the initial page that is generated as the response from the Forms servlet, to ensure the reusability of the applet when `legacy_lifecycle` is set to true. When set to true, JavaScript must be enabled on the Java client. Default value is `false`.
`maxRuntimeProcesses`	Optional	This specifies the maximum allowable number of concurrent Forms run-time processes. It should be set a value that reflects the customer's hardware configuration (and the portion that can be used by Forms applications). A value of 0 (the default) indicates that there is no explicit limit. This default is not recommended, because it leaves the system vulnerable to Denial of Service attacks. Default value if not specified is 0.
`networkRetries`	Optional	Number of times client should retry if a network failure occurs. Default value is 0.
`obr`	Optional	For internal use only. Default value is no. This parameter is a sub-argument for otherparams.
`otherparams`	Optional	This setting specifies command line parameters to pass to the Forms run-time process in addition to form and userid. This parameter is a runform parameter. Default value from `formsweb.cfg` is `obr=%obr% record=%record% tracegroup=%tracegroup% log=%log% term=%term% ssoProxyConnect=%ssoProxyConnect%` **Note**: Special syntax rules apply to this parameter when it is specified in a URL: a + may be used to separate multiple name=value pairs (see Section 3.3.4, "Specifying Special Characters in Values of Runform Parameters" for more information). For production environments, to provide better control over which runform parameters, end users can specify in a URL, include the otherparams parameter in the value of the restrictedURLparams parameter.

Table 4–13 (Cont.) Advanced Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
prestartIncrement	Optional	The number of run-time processes to be created when the number of prestarted run-time processes is less than minRuntimes. Default value if not specified is 1.
prestartInit	Optional	Number of the run-time processes that should be spawned initially. Default value if not specified is 1.
prestartMin	Optional	Minimum number of run-time processes to exist in the pool. Default value if not specified is 0.
prestartRuntimes	Optional	Run-time prestarting or pooling is enabled only if true. Default value if not specified is false.
prestartTimeout	Optional	Time in minutes after which all the prestarted processes of this pool (configuration section) is stopped. A run-time process is removed from the prestart pool after the client connection is made and thus is not stopped. Default value if not specified is 0.
query_only	Optional	Set this parameter to yes to prevent the end user from inserting, updating, or deleting records. Default value if not specified is no. This parameter is a sub-argument for otherparams.
quiet	Optional	Set this parameter to yes to prevent messages from producing an audible beep. Default value if not specified is no. This parameter is a sub-argument for otherparams.
recordFileName	Optional	Forms applet parameter that specifies the name of file (for example, d:\temp\is) that stores the recorded Forms messages. Default value if not specified is 'is' (without the quotes).
restrictedURLparams	Optional	Forms applet parameter. Specifies a comma-delimited list of parameters which is rejected if specified in a URL. Default value from formsweb.cfg is "pageTitle,HTMLbodyAttrs,HTMLbeforeForm,HTMLafterForm,log".
restrictedURLchars	Optional	Forms applet parameter. Specifies a comma-delimited characters that is restricted for use in the request URL's query string.
serverApp	Optional	Forms applet parameter. Default value is default.
serverURL	Required	Determines the URL path to Forms Listener Servlet. Default value is /forms/lservlet.

Table 4–13 (Cont.) Advanced Configuration Parameters

Parameter	Required/ Optional	Parameter Value and Description
term	Optional	The full path of a custom key binding file (to be used instead of the standard fmrweb or fmrweb_utf8 files). This parameter is a sub-argument for otherparams.

4.2.5.8 List of Parameters that Cannot be Specified in the URL

This section lists the parameters that can be specified only in the servlet configuration file (formsweb.cfg). If any are specified in the URL, the value is ignored. In addition, any parameter that is listed in the value of the restrictedURLparams parameter is rejected if specified in the URL.

- allowNewConnections
- baseHTML
- baseHTMLjpi
- connectionDisallowedURL
- defaultCharset
- envFile
- escapeparams
- HTMLdelimiter
- maxRuntimeProcesses
- prestartIncrement
- prestartInit
- prestartMin
- prestartRuntimes
- prestartTimeout
- restrictedURLparams
- restrictedURLchars
- serverURL
- ssoCancelURL
- ssoDynamicResourceCreate
- ssoErrorURL
- ssoMode
- workingDirectory

4.3 Managing Environment Variables

Use the **Environment Configuration** page of Fusion Middleware Control to manage environment variables. From this page, you can add, edit, or delete environment variables as necessary.

The environment variables such as `PATH`, `ORACLE_INSTANCE`, `ORACLE_HOME`, and `FORMS_PATH` for the Forms run-time executable (`frmweb.exe` on Windows and `frmweb` on UNIX) are defined in `default.env`. The Forms listener servlet calls the executable and initializes it with the variable values provided in the environment file, which is found in the `$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config` directory by default.

Any environment variable that is not defined in `default.env` is inherited from the Oracle WebLogic Managed Server. The environment file must be named in the `envFile` parameter in the Default section of the **Web Configuration** page.

A few things to keep in mind when customizing environment variables are:

- Environment variables may also be specified in the Windows registry. Values in the environment file override settings in the registry. If a variable is not set in the environment file, the registry value is used.
- You need administrator privileges to alter registry values.
- The server does not require restarting for configuration changes to take effect.
- Existing Forms processes are not affected by environment variables that were defined after they were started.
- Environment variables not set in the environment file or Windows registry are inherited from the environment of the parent process, which is the Oracle WebLogic Managed Server.

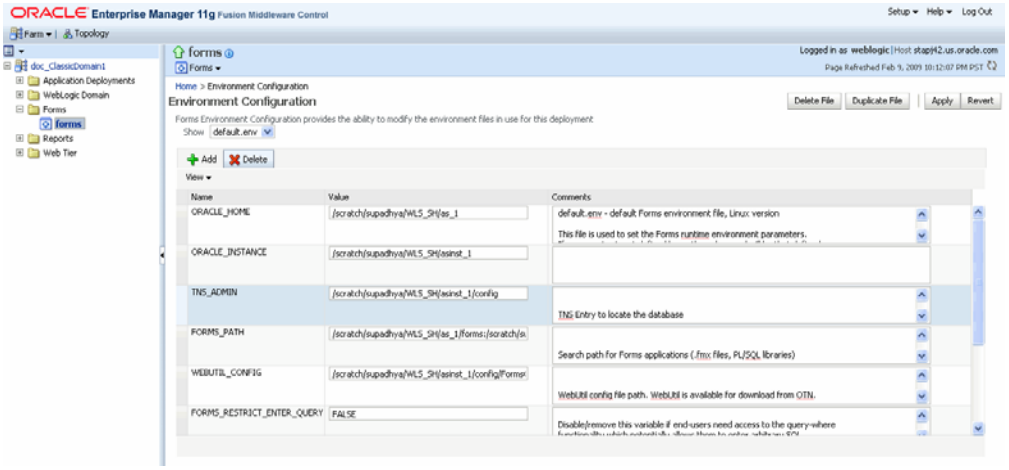
Table 4–14, " Default Environment Variables" describes important environment variables that are specified in `default.env`.

4.3.1 Managing Environment Configuration Files

To access the Environment Configuration page:

1. Start Fusion Middleware Control.
2. From the Fusion Middleware Control main page, click the link to the Oracle Forms Services instance that you want to configure.
3. From the Forms menu list, select **Environment Configuration.** The **Environment Configuration** page (Figure 4–4) is displayed.

Figure 4–4 Environment Configuration page



To duplicate an environment configuration file:

1. From the **Environment Configuration** page, click Duplicate File.

   The Duplicate File dialog is displayed.
2. Select the file which you want to duplicate and enter a unique name for the file.
3. Click Duplicate to create the file.

To delete an environment configuration file:

1. In the **Environment Configuration** page, from the **Show** menu list, select the environment configuration file you want to delete.
2. Click **Delete File**.

   The Confirmation dialog is displayed.
3. Click **Yes** to confirm the deletion.

> **Note:** You cannot delete `default.env`. You can delete only user-defined environment configuration files.

To view an environment configuration file:

1. In the **Environment Configuration** page, from the Show menu list, select the environment configuration file that you want to view.
2. The parameters and their values are displayed.

4.3.2 Configuring Environment Variables

To edit an environment variable:

1. In the **Environment Configuration** page, select the row of the parameter that contains the environment variable you want to edit.
2. Enter the Value and Comments.
3. Click **Apply** to save the changes or **Revert** to discard them.

To add an environment variable:

1. From the **Show** menu list, select the environment configuration file to which you want to add the variable.
2. Click **Add** to add a parameter.

   The Add dialog box is displayed.
3. Enter the Name, Value and Comments.
4. Click **Create**.
5. Click **Apply** to save the changes or **Revert** to discard them.

To delete an environment variable:

1. From the **Show** menu list, select the environment configuration file where you want to delete an environment variable.
2. Select the rows of the parameters you want to delete. You can delete more than one parameter at a time.

3. Click **Delete**.
4. Click **Apply** to save the changes or **Revert** to discard them.

4.3.3 Default Environment Variables

Table 4–14 provides the valid values and a description of some of the environment variables.

Table 4–14 Default Environment Variables

Parameter	Valid Values	Description
ORACLE_HOME	ORACLE_HOME (default)	Points to the base installation directory of any Oracle product.
ORACLE_INSTANCE	ORACLE_INSTANCE (default)	Contains all configuration files, repositories, log files, deployed applications, and temporary files.
PATH	ORACLE_HOME/bin (default)	Contains the executables for Oracle products.
FORMS_PATH	ORACLE_HOME/forms:ORACLE_INSTANCE/FormsComponent/forms (default)	Specifies the path that Oracle Forms searches when looking for a form, menu, or library to run. For **Windows**, separate paths with a *semi-colon* (;). For **UNIX**, separate paths with a *colon* (:).
FORMS_RESTRICT_ENTER_QUERY	TRUE (default)	Disable or remove this variable for end-users who need access to the query-where functionality which potentially allows them to enter arbitrary SQL statements when in enter-query mode.
TNS_ADMIN	ORACLE_INSTANCE/config	Specifies the path name to the TNS files such as TNSNAMES.ORA, SQLNET.ORA and so on.
CLASSPATH	ORACLE_HOME/jdk/bin/java	Specifies the Java class path, which is required for the Forms debugger.

Table 4–14 (Cont.) Default Environment Variables

Parameter	Valid Values	Description

LD_LIBRARY_PATH	Set the LD_LIBRARY_PATH environment variable for the first time to
	ORACLE_HOME/lib.
	You can reset LD_LIBRARY_PATH in the Bourne shell by entering:
	`$ set LD_LIBRARY_PATH=ORACLE_HOME/lib:${LD_LIBRARY_PATH}`
	`$ export LD_LIBRARY_PATH`
	or in the C shell by entering:
	`% setenv LD_LIBRARY_PATH ORACLE_HOME/lib:${LD_LIBRARY_PATH}`
		Oracle Forms Developer and Reports Developer products use dynamic, or shared, libraries. Therefore, you must set LD_LIBRARY_PATH so that the dynamic linker can find the libraries.

WEBUTIL_CONFIG	ORACLE_INSTANCE/config/FormsComponent/forms/server/webutil.cfg	
FORMS_MESSAGE_ENCRYPTION	TRUE	Possible values are TRUE or FALSE. Use this environment variable to turn off or on the proprietary obfuscation applied to Forms messages when using HTTP mode. By default, communication is obfuscated.
LD_PRELOAD	<JDK_HOME>/jre/lib/i386/libjsig.so	Specifies the location of the library libjsig.so. This library is used for the signal-chaining facility offered by JVM 1.5. The signal-chaining facility enables an application to link and load the shared library libjsig.so before the system libraries. Ensure this is set for Forms and Reports integration on UNIX/Linux. **Note**: If there are multiple environment files, ensure that LD_PRELOAD has the same settings as in default.env.
FORMS_PLSQL_BHVR_COMMON_SQL	Set the value of FORMS_PLSQL_BHVR_COMMON_SQL to true or 1 to enable the feature. Set the value of to false or 0 to disable the feature.	If this variable is set, then PL/SQL uses a common SQL parser (that is, the one in RDBMS SQL engine) for compiling SQL code rather than the separate one built in to PL/SQL used for compiling static SQL.

Note: On Windows, Oracle Forms Services reads Oracle environment settings from the Windows Registry unless they are set as environment variables.

4.4 Managing User Sessions

Administrators can manage user sessions, and related features such as monitoring, debugging and tracing using Fusion Middleware Control.

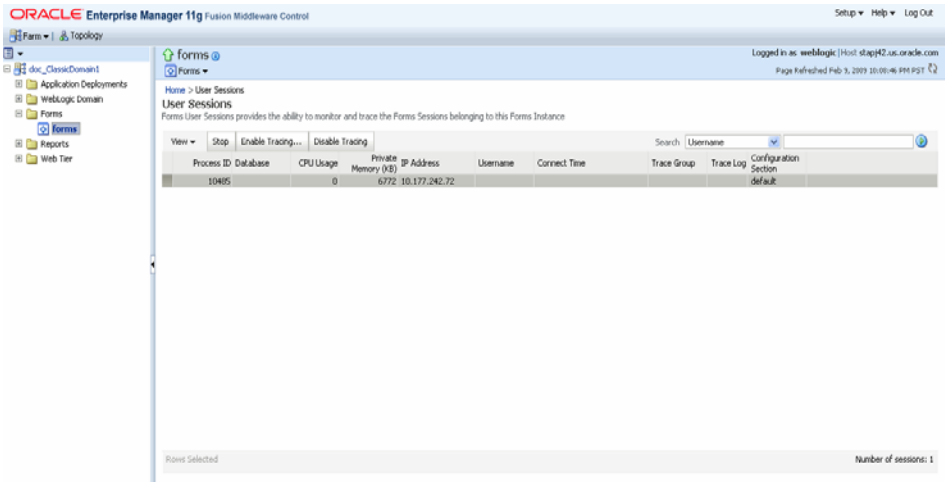
A user session starts when the frmweb process starts. Use the Forms User Sessions pages to monitor and trace the Forms sessions within a Forms Instance. The Forms User Sessions page is accessed from the Forms menu list by selecting **User Sessions**.

To view Forms user sessions:

1. Start Fusion Middleware Control.
2. From the Forms menu list, select **User Sessions**.

   The **User Sessions** page (Figure 4–5) is displayed.

Figure 4–5 User Sessions page



3. Table 4–15 describes the fields on the User Sessions page.

Table 4–15 User Sessions Page

Field	Description
Process ID	The process ID of the user session.
Database	The database name used by the Forms application for the user session. Click the Database name to view the Database Sessions page.
CPU Usage	The percentage of CPU used by the run-time process.
Private Memory (KB)	The memory used by the run-time process. On Linux platforms, private memory is not the actual private memory but indicates the Resident Set Size (RSS).
IP Address	The IP address of the client computer used to connect to Forms Services.
Username	Database user name.

Table 4–15 (Cont.) User Sessions Page

Field	Description
Connect Time	The time when the user connected to Forms Services. If the client connection time and client IP are empty, the session is a prestarted session, which is not yet connected to any client.
Trace Group	The trace group used for tracing the user session. When tracing is enabled, this column shows the trace group name or the events being traced. The events are displayed if the events of the trace group that was enabled for the session have been later modified in the trace configuration. Note that the Trace group name that is displayed may not be indicate the accurate events being traced if built-ins are used to control the tracing.
Trace Log	Displays the trace log if one exists for the user session.
Configuration Section	Indicates the configuration section used by the Forms application.
Form Name	Indicates the module name of the form application.
CPU Time	Indicates total CPU time used by forms sessions since Connect time.

To enable new Forms user sessions:

By default, new Forms user sessions are enabled. You can disable them by using Fusion Middleware Control to set the `allowNewConnections` parameter to false.

1. Start Fusion Middleware Control.
2. From the Forms menu, select **Web Configuration**.
3. Select the default configuration section. allowNewConnections cannot be overridden in named sections.
4. In the Sections region, find and edit the value for the `allowNewConnections` parameter. A value of `true` (default) enables new user sessions, whereas `false` disables them.
5. Click **Apply** to save the changes.

To disable new Forms user sessions:

1. Start Fusion Middleware Control.
2. From the Forms menu, select **Web Configuration**.
3. Select the default configuration section. allowNewConnections cannot be overridden in named sections.
4. In the Sections region, find and edit the value for the `allowNewConnections` parameter. A value of `true` (default) enables new user sessions, whereas `false` disables them.
5. Click **Apply** to save the changes.

When new user sessions are disabled, attempted connections are directed to a URL identified by the `formsweb.cfg` parameter `connectionDisallowedURL` (in the default section). You must specify a complete and valid URL as the value.

If `connectionDisallowedURL` is not specified, then the following message is displayed in the browser:

```
The Forms servlet will not allow new connections. Please contact
your System Administrator.
```

When you disable new user sessions, existing forms sessions are unaffected and the Oracle WebLogic Managed Server instance remains up.

To enable tracing for a Forms user sessions:

1. Start Fusion Middleware Control.
2. In the User Sessions page, select the row that has the user session for which you want to enable tracing.
3. Select Enable Tracing.
4. From the Select Trace Group list, select an available trace group and click **OK**.

To disable tracing for a Forms user sessions:

1. In the User Sessions page, select the row that has the user session for which you want to disable tracing.
2. Click **Disable Tracing**.
3. Click **OK**. The Disable Tracing dialog is dismissed and tracing is now stopped for the selected Forms user session.

To terminate a Forms user session:

1. Select the link to the Forms Services instance that has the user session to be terminated.
2. From the Forms menu, select **User Sessions**.
3. Click the row of the user session to be deleted.
4. Click **Stop**.
5. The Confirmation dialog is displayed.
6. Click **Yes**.

   The user session is deleted and the Runform instance is terminated.

To view trace logs of a Forms user sessions:

1. From the Forms menu, select **User Sessions**.
2. For a user session that is active, click **View Trace Log** in the **Trace Log** column. Log in to view the trace file.

To search for a Forms user sessions:

1. From the Forms menu, select **User Sessions**.
2. Select the column name in which you want to search.
3. Enter the search string.
4. Click the blue arrow to search. The search results are displayed.

To sort the list of Forms user sessions:

1. From the Forms menu, select **User Sessions**.
2. Move the mouse over the column.

3. Click the up or down arrow to sort in ascending or descending order. The page is refreshed showing the sorted user sessions. You can sort in order of all columns except Trace Logs.

To customize your view of Forms user sessions:

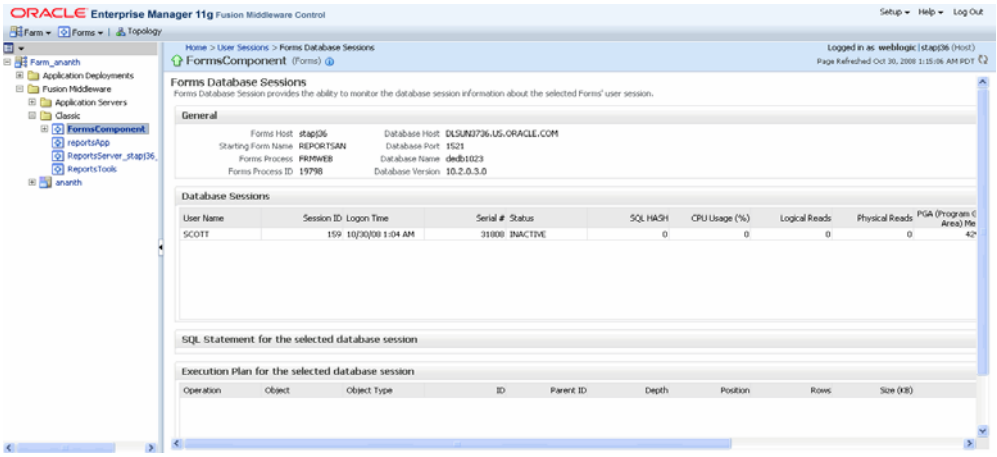
1. From the User Sessions page, click View.
2. From the View menu, you can:
   - Select **Show All** to view all columns.
   - Select specific columns you want displayed.
   - Select **Reorder Columns** to organize the order of display of the columns.
   - Select **Show More Columns** to hide or display specific columns.

To view database sessions for a Forms user session:

1. From the Forms menu, select **User Sessions**.
2. Click the Database name in the Database column.

   Log in to view the Database Sessions page (Figure 4–6). You need Database Administrator privileges to log in to Database Sessions page.

Figure 4–6 Database Sessions Page



3. Table 4–16, Table 4–17, and Table 4–18 describe the information displayed in the Database Sessions page.

Table 4–16 Database Sessions Page

Field	Description
Username	Database username used for connection to the database.
Session ID	Database session identifier.
Logon Time	Date and time when user logged on to the session.
Serial #	Session serial number. Used to uniquely identify a session's objects. Guarantees that session-level commands are applied to the correct session objects if the session ends and another session begins with the same session ID.
Status	Indicates whether the session is active or not.

Table 4–16 (Cont.) Database Sessions Page

Field	Description
SQL HASH	Used to identify the SQL statement executed
CPU Usage (%)	CPU Usage (in percentage) on the Database system for the given session.
Logical Reads	Number of Logical Reads for the given session.
Physical Reads	Number of Physical Reads for the given session.
PGA (Program Global Area) Memory	Size of PGA (Program Global Area) Memory after an interval.

Table 4–17 Details of Selected Database Session

Field	Description
SQL Statement for the selected Database Session	Displays the most recent SQL statement.

Table 4–18 Execution Plan for the Selected Database Session

Field	Description
Operation	Name of the internal operation performed in the execution step (for example, TABLE ACCESS).
Object	Name of the table or index.
Object Type	Type of the object.
ID	A number assigned to each step in the execution plan.
Parent ID	ID of the next execution step that operates on the output of the current step.
Depth	Depth (or level) of the operation in the tree. It is not necessary to issue a CONNECT BY statement to get the level information, which is generally used to indent the rows from the PLAN_ TABLE table. The root operation (statement) is level 0.
Position	Order of processing for all operations that have the same PARENT_ID.
Rows	Estimate, by the cost-based optimizer, of the number of rows produced by the operation.
Size (KB)	Estimate, by the cost-based optimizer, of the number of bytes produced by the operation.
Cost	Cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.
Time (sec)	Elapsed time (in seconds) of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.
CPU Cost	CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.
I/O Cost	I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.

4.5 Managing URL Security for Applications

Oracle Forms applications are web-deployed solutions that users access through a browser. Oracle Forms architecture allows Forms developers two ways to choose and configure how a Forms application runs. One option is to set the parameter and the value in the URL. The second option is to set the parameter and its value(s) in the configuration file, that is, `formsweb.cfg`. The parameter that is set in the `formsweb.cfg` can be overridden by the parameter set in the URL.

A Forms administrator can override this default behavior, and give the Forms administrator full control over what parameter can be used in the URL.

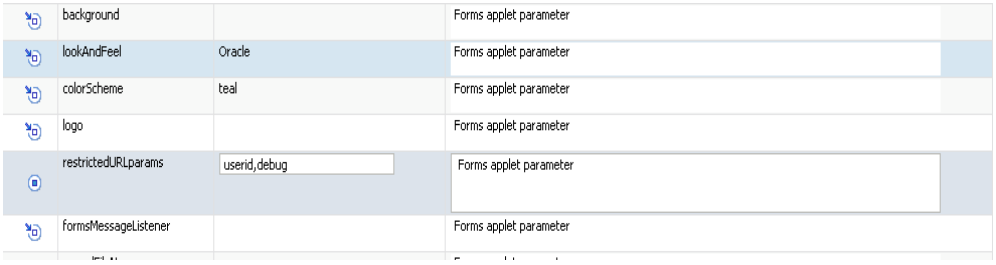
Here are two scenarios to consider when deciding which parameters to allow or not allow in a URL. The first scenario is when an administrator just wants to restrict the usage of the USERID parameter in the URL that forces the end-user to always log in using the default login window. The second scenario is when an administrator disables all parameters except a few, such as `CONFIG=MyApp` in a URL.

The parameter `restrictedURLparams` allows flexibility for the Forms administrator to consider any URL-accessible parameter in the `formsweb.cfg` file as restricted to a user. An administrator can specify this parameter in a named configuration section to override the one specified in the default configuration section. The `restrictedURLparams` parameter itself cannot be set in the URL.

By design, command line arguments passed in a URL always override similar definitions in the formsweb.cfg.

In this example, the `userid` is defined as `scott/tiger` and `debug` is set to `false`. An application that is configured to connect to the database as `scott/tiger` can connect as a different user with the `userid` parameter added as a URL parameter. To prevent this, the `userid` parameter is defined in the `restrictedURLparams` as shown in Figure 4–7, "Defining the restrictedURLparams Parameter".

Figure 4–7 Defining the restrictedURLparams Parameter



Similarly, an administrator can use the `restrictedURLparams` parameter to redirect a user to a page which lists the restricted parameters that were used.

4.5.1 Securing the Oracle Forms Test Form

The test form runs when you access an Oracle Forms URL but do not specify an application to run. For example, normally you call an Oracle Forms application with the following syntax:

```
http://<host>:<port>/forms/frmservlet?config=myApp
```

The Forms servlet locates `[myApp]` in the formsweb.cfg file and launches that application. However, when no application is specified, for example:

```
http://<host>:<port>/forms/frmservlet
```

The Forms servlet uses the settings in the default section of the formsweb.cfg file. These settings are located under `[default]` in the Forms Configuration file (anytime an application does not override any of these settings, the defaults are used). The default section has the following setting:

```
form=test.fmx
```

This is the test form which enables you to test your Oracle Forms Services installation and configuration. Thus if you do not specify an application, Forms launches the test.fmx file. You could change this to:

```
form=
```

And the form does not run. However, this is not optimal; the Forms servlet still sends the dynamically generated HTML file to the client, from which a curious user could obtain information. The optimally secure solution is to redirect requests to an informational HTML page that is presented to the client instead. Some parameters in the `formsweb.cfg` file must be changed.

Here are the parameters to change, along with their default values when you install Oracle Forms Services:

```
# System parameter: default base HTML file
baseHTML=base.htm
# System parameter: base HTML file for use with Sun's Java Plug-In
baseHTMLjpi=basejpi.htm
```

These parameters are templates for the HTML information that are sent to the client. Create an informational HTML page and have these variables point to that instead. For example, in the `$ORACLE_INSTANCE/config/FormsComponent/forms/server` directory, create a simple HTML page called `forbidden.html` with the following content:

```
<html>
  <head>
    <title>Forbidden</title>
  </head>
  <body>
   <h1>Forbidden!</h1>
    <h2>You may not access this Forms application.</h2>
  </body>
</html>
```

> **Note:** This message page displayed as a result of redirecting of client information is different from the page that the Web server returns when the requested content has restricted permissions on it.

Next, modify the `formsweb.cfg` parameters by commenting out or modifying the original parameters:

```
# System parameter: default base HTML file
#baseHTML=base.htm
baseHTML=forbidden.html
# System parameter: base HTML file for use with Sun's Java Plug-In
#baseHTMLjpi=basejpi.htm
baseHTMLjpi=forbidden.html
# System parameter: base HTML file for use with Microsoft Internet Explorer
# (when using the native JVM)
```

When a user enters the URL

```
http://<host>:<port>/forms/frmservlet
```

the customized Web page is presented. Of course, you can customize `forbidden.html`, including its contents, its filename, and its location if you make the corresponding changes to these parameters in the `formsweb.cfg` file. Administrators can put any information, such as warnings, errors, time stamps, IP logging, or contact information in this information Web page with minimal impact on the server configuration.

Note: Overriding the base HTML template entries in the default section of `formsweb.cfg` requires that you add the same entries pointing to the original values (or some other valid HTML file) in your application-specific named configuration:

```
[myApp]
form=myApplication.fmx
lookandfeel=oracle
baseHTML=base.htm
baseHTMLjpi=basejpi.htm
```

If you do not specify these base HTML values, and when a user runs an application, the `forbidden.html` page is displayed because the application-specific configuration section has not overridden the default values.

4.6 Creating Your Own Template HTML Files

Consider creating your own HTML file templates (by modifying the templates provided by Oracle). By doing this, you can hard-code standard Forms parameters and parameter values into the template. Your template can include standard text, a browser window title, or images (such as a company logo) that would appear on the first Web page users see when they run Web-enabled forms. Adding standard parameters, values, and additional text or images reduces the amount of work required to customize the template for a specific application. To add text, images, or a window title, you must include the appropriate tags in the template HTML file.

See Chapter 3.3.4, "Specifying Special Characters in Values of Runform Parameters" for information about coding the `serverArgs` applet parameter.

Any user-added customized configuration files (such as user client registry files or user key binding files or multiple environment files) must be copied to the same directory as the corresponding default configuration file.

For example, if the user has created a French environment configuration file `default_fr.env`, then it must be placed in the `$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config` directory.

4.6.1 Variable References in Template HTML Files

When a variable reference occurs within a string delimited by quotes or apostrophes (for example, the value of an applet parameter), then when the value of the variable is substituted for the variable reference, HTML metacharacters ('&', '<', '>', quote, and apostrophe) are replaced by HTML escape sequences.

This sequence is *not* done for variable references outside delimited strings. Therefore, such variables should be specified in the `restrictedURLparams` system default configuration parameter, for security reasons.

Note: To modify the cursor blink rate, or disable blinking, set the client parameter `cursorBlinkRate` as follows.

```
<PARAM NAME="cursorBlinkRate" VALUE="1000">
```

The default is 600 milliseconds: the cursor completes one full blink every 1.2 seconds (1200 ms).

A value of zero disables the blinking and the cursor remains visible all the time.

4.7 Deploying Fonts, Icons, and Images Used by Forms Services

This section explains how to specify the default location and search paths for fonts, icons, and images in Registry.dat. To look at a sample of the default Registry.dat file, see Section C.8.1, "Registry.dat".

4.7.1 Managing Registry.dat with Fusion Middleware Control

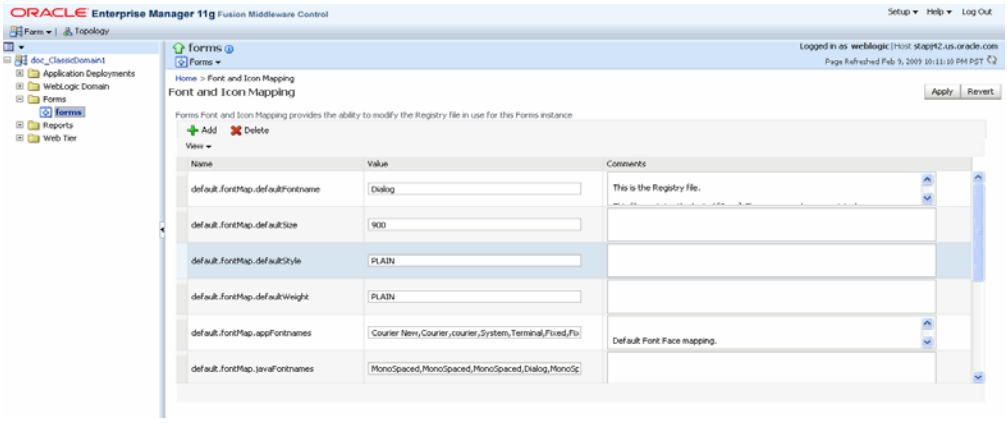
Use Fusion Middleware Control to change, add, or delete parameters from `Registry.dat`.

To access the Fonts and Icon Mapping page:

1. Start Fusion Middleware Control.
2. From the Forms menu list, select **Font and Icon Mapping**.

   The Font and Icon Mapping page (Figure 4–8) is displayed.

Figure 4–8 Font and Icon Mapping Page



To edit a Registry.dat parameter value:

1. Start Fusion Middleware Control.
2. From the Forms menu list, select **Font and Icon Mapping**.
3. Select the row containing the parameter to modify and change the value(s) for it in the **Value** text field.

4. Click **Apply** to save the changes.

To add a Registry.dat parameter and its value:

1. From the Forms menu list, select **Font and Icon Mapping**.
2. Click Add.

   The Add dialog appears.
3. Enter the name, value, and comments for this parameter.
4. Click **Create**.
5. Click **Apply** to save or **Revert** to discard the changes.

To delete a Registry.dat parameter and its value:

1. From the Forms menu list, select **Font and Icon Mapping**.
2. Select the row containing the parameter to delete and click **Delete**.
3. The parameter is deleted.
4. Click **Apply** to save or **Revert** to discard the changes.

4.7.2 Managing Application Fonts

Using Fusion Middleware Control, you can also change the default font and font settings by the Registry.dat file. All font names are Java Font names. Each of these parameters represents the default property to use when none is specified.

To change the font settings for a deployed application:

1. Start Fusion Middleware Control.
2. From the Forms menu list, select **Font and Icon Mapping**.
3. Change any of the settings to reflect your desired font setting, based on Table 4–19:

Table 4–19 Default Font Values

Font Name	Default Value
default.fontMap.defaultFontname	Dialog Represents the default Java fontName.
default.fontMap.defaultSize	900 Represents the default fontSize. Note that the size is multiplied by 100 (for example, a 10pt font has a size of 1000).
default.fontMap.defaultStyle	PLAIN Represents the default fontStyle, PLAIN or *ITALIC*.

Table 4–19 (Cont.) Default Font Values

Font Name	Default Value
default.fontMap.defaultWeight	PLAIN Represents the default fontWeight, PLAIN or **BOLD**.
default.fontMap.appFontnames	Courier New,Courier,courier,System,Terminal,Fixedsys,Times,Times New Roman,MS Sans Serif,Arial Default Font Face mapping. Represents a comma delimited list of application font names. The number of entries in the appFontname list should match the number in the javaFontname list. The elements of the list are comma separated and *all* characters are taken literally; leading and trailing spaces are stripped from Face names. Note that this file uses the Java 1.1 font names to handle the NLS Plane.
default.fontMap.javaFontnames	MonoSpaced,MonoSpaced,MonoSpaced,Dialog,MonoSpaced,Dialog,Dialog,Serif,Serif,Dialog,SansSerif Represents a comma delimited list of Java font names.

For example, to change your default font to Times New Roman, replace **Dialog** with **Times New Roman**.

You can change the default font face mappings:

```
default.fontMap.appFontnames=Courier New,Courier,
courier,System,Terminal,Fixed,Fixedsys,Times,Times New Roman,
MS Sans Serif,Arial
default.fontMap.javaFontnames=MonoSpaced,MonoSpaced,MonoSpaced,Dialog,
MonoSpaced,Dialog,Dialog,Serif,Serif,Dialog,SansSerif
```

4. Click **Apply** to save the changes.

Some fonts on Windows are not supported in Java. For this reason you can specify (map) Java-supported fonts that appear when a non-supported font is encountered. In the previous sample, each font in default.fontMap.appFontnames corresponds to a font in default.fontMap.javaFontnames.

4.7.3 Deploying Application Icons

When deploying an Oracle Forms application, the icon files used must be in a Web-enabled format, such as JPG or GIF (GIF is the default format).

By default, the icons are found relative to the `DocumentBase` directory. That is, `DocumentBase` looks for images in the directory relative to the base directory of the application start HTML file. As the start HTML file is dynamically rendered by the Forms servlet, the `forms` directory becomes the document base.

For example, if an application defines the icon location for a button with `myapp/<iconname>`, then the icon is looked up in the directory `forms/myapp`.

To change the default location, set the `imageBase` parameter to `codebase` in the Web Configuration page of Enterprise Manager Fusion Middleware Control. Alternatively, you can change the `default.icons.iconpath` value of the Registry.dat file in the `$DOMAIN_HOME/config/fmwconfig/servers/WLS_`

`FORMS/applications/formsapp_11.1.1/config/forms/registry/oracle/forms/registry` directory.

Setting the imageBase parameter to `codebase` enables Oracle Forms to search the `forms/java` directory for the icon files. Use this setting if your images are stored in a Java archive file. Changing the image location in the Registry.dat configuration file is useful to store images in a central location independent of any application and independent of the Oracle Forms installation.

4.7.3.1 Storing Icons in a Java Archive File

If an application uses a lot of custom icon images, it is recommended you store icons in a Java archive file and set the `imageBase` value to `codebase`. The icon files can be zipped to a Java archive using the Jar command of any Java Software Development Kit (Java SDK).

For example, the command `jar -cvf myico.jar *.gif` packages all files with the extension `.gif` into an archive file with the name myico.jar.

In order for Oracle Forms to access the icon files stored in this archive, the archive must be stored into the `forms/java` directory. Also, the name of the archive file must be part of the archive tag used in the custom application section of the `formsweb.cfg` file. Now, when the initial application starts, the icon files are downloaded and permanently stored on the client until the archive file is changed.

Note: Oracle Forms default icons (for example, icons present in the default smart icon bar) do not require deployment, as they are part of the `frmall.jar` file.

4.7.3.2 Adding, Modifying, and Deleting Icon Mappings

Use Fusion Middleware Control to add icon changes to the Registry.dat file used by your application.

To add icon mappings:

1. Start Fusion Middleware Control.
2. From the Forms menu, select **Font and Icon Mapping**.
3. Click **Add**.

   The Add dialog appears.
4. Enter the name, value, and an optional comment.
5. Click **Create** to create the mapping.

   The mapping is added to the list.
6. Click **Apply** to save the changes.

To modify icon mappings:

1. From the Font and Icon Mapping region, select the mapping you want to modify.
2. Change the name and value of the mapping. For example,
   - Modify the `iconpath` parameter specifying your icon location:

     `default.icons.iconpath=/mydir`

     (for an absolute path)

or

`default.icons.iconpath=mydir`

(for a relative path, starting from the `DocumentBase` Directory)

- Modify the `iconextension` parameter:

  `default.icons.iconextension=gif`

  or

  `default.icons.iconextension=jpg`

3. Click **Apply** to save and activate the changes.

To delete an icon mapping:

1. From the Font and Icon Mapping region, select the mapping you want to delete.
2. Click **Delete**.
3. The selected icon mapping is deleted.
4. Click **Apply** to save or **Revert** to discard the changes.

To reference the application file:

- In a specific named configuration section in the `formsweb.cfg` file, modify the value of the `serverApp` parameter and set the value to the location and name of your application file.

  For example:

  ```
  [my_app]
  ServerApp=/appfile/myapp
  ```

  (for an absolute path)

  or

  ```
  [my_app]
  ServerApp=appfile/myapp
  ```

  (for a relative path, relative to the `CodeBase` directory)

  Table 4–20 describes the correct locations where to place your application icons:

Table 4–20 Icon Location Guide

Icon Location	When	How
DocumentBase	Default. Applications with few or no custom icons.	Store icons in forms directory or in a directory relative to forms.
Java Archives	Applications that use many custom icons.	Set `ImageBase` to `codebase`, create Java archive file for icons, and add archive file to the archive parameter in `formsweb.cfg`.

Table 4–20 (Cont.) Icon Location Guide

Icon Location	When	How
Registry.dat	Applications with custom icons that are stored in a different location as the Oracle Forms install (can be another server). Useful to make other changes to the Registry.dat file such as font mapping.	Copy Registry.dat and change ServerApp parameter in `formsweb.cfg`.

4.7.4 Splash screen and Background Images

When you deploy your applications, you have the ability to specify a splash screen image (displayed during the connection) and a background image file.

Those images are defined in the HTML file or you can use the **Web Configuration** page in Enterprise Manager:

```
<PARAM NAME="splashScreen" VALUE="splash.gif">
```

```
<PARAM NAME="background" VALUE="back.gif">
```

The default location for the splash screen and background image files is in the DocumentBase directory containing the baseHTML file.

Note: Image formats for splash screens and icons are the standard formats that are supported by `java.awt.Image`. For more information on `java.awt.Image`, refer to the Java Advanced Imaging (JAI) API at `http://java.sun.com`.

4.7.5 Custom Jar Files Containing Icons and Images

Each time you use an icon or an image (for a splash screen or background), an HTTP request is sent to the Web server. To reduce the HTTP round-trips between the client and the server, you have the ability to store your icons and images in a Java archive (Jar) file. Using this technique, only one HTTP round-trip is necessary to download the Jar file.

4.7.5.1 Creating a Jar File for Images

The Java SDK comes with an executable called *jar*. This utility enables you to store files inside a Java archive. For more information, see `http://java.sun.com/`.

For example:

```
jar -cvf myico.jar Splash.gif Back.gif icon1.gif
```

This command stores three files (`Splash.gif`, `Back.gif`, `icon1.gif`) in a single Jar file called `myico.jar`.

4.7.5.2 Using Files Within the Jar File

The default search path for the icons and images is relative to the documentBase. However, when you want to use a Jar file to store those files, the search path must be relative to the `codebase` directory, the directory which contains the Java applet.

To use a Jar file to store icons and images, you must specify that the search path is relative to `codebase` using the imageBase parameter in the `formsweb.cfg` file or HTML file.

This parameter accepts two different values:

- **documentBase** The search path is relative to the `documentBase` directory. If no value is specified for imageBase, then the value of documentBase is used.
- **codeBase** The search path is relative to the `codeBase` directory, which gives the ability to use Jar files.

In this example, we use a JAR file containing the icons and we specify that the search should be relative to codeBase. If the parameter `imageBase` is not set, the search is relative to `documentBase` and the icons are not retrieved from the Jar file.

For example (`formsweb.cfg`):

```
archive=frmall.jar, icons.jar

imageBase=codeBase
```

4.7.6 Search Path for Icons and Images

The icons and images search path depends on:

- What you specify in your custom application file (for the icons).
- What you specified in the `splashScreen` and `background` parameters of your default Forms configuration file or HTML file (for the images).
- What you specify in the `imageBase` parameter in the **Web Configuration** page of Fusion Middleware Control for the file or HTML file (for both icons and images).

Forms Services searches for the icons depending on what you specify. This example assumes:

- *host* is the computer name.
- `DocumentBase` is the URL pointing to the HTML file.
- codebase is the URL pointing to the location of the starting class file (as specified in the `formsweb.cfg` file or HTML file).
- *mydir* is the URL pointing to your icons or images directory.

4.7.6.1 DocumentBase

The default search paths for icons and images are relative to the `DocumentBase`. In this case, do not specify the imageBase parameter:

Table 4–21 Search Paths for Icons

Location Specified	Search path used by Forms Services
default	http://host/documentbase
`iconpath=mydir` (specified in your application file)	`http://host/documentbase/mydir` (relative path)

Table 4–21 (Cont.) Search Paths for Icons

Location Specified	Search path used by Forms Services
`iconpath=/mydir` (specified in your application file)	`http://host/mydir` (absolute path)

Table 4–22 Search Paths for Images

Location Specified	Search Path Used by Forms Services
file.gif (specified, for example, in formsweb.cfg as splashscreen=file.cfg)	http://host/documentbase/file.gif
mydir/file.gif	`http://host/documentbase/mydir/file.gif` (relative path)
/mydir/file.gif	`http://host/mydir/file.gif` (absolute path)

4.7.6.2 codebase

Use the `imageBase=codebase` parameter to enable the search of the icons (Table 4–23) and images (Table 4–24) in a Jar file:

Table 4–23 Icon Search Paths Used by Forms Services

Location Specified	Search Path Used by Forms Services
default	`http://host/codebase` or root of the Jar file
`iconpath=mydir` (specified in your application file)	`http://host/codebase/mydir` or in the mydir directory in the Jar file (relative path)
`iconpath=/mydir` (specified in your application file)	`http://host/mydir` (absolute path) No Jar file is used.

Table 4–24 Image Search Paths Used by Forms Services

Location Specified	Search Path Used by Forms Services
file.gif	`http://host/codebase/file.gif` or root of the Jar file
`mydir/file.gif` (specified in your HTML file)	`http://host/codebase/mydir/file.gif` or in the mydir directory in the Jar file (relative path)
`/mydir/file.gif` (specified in your HTML file)	`http://host/mydir/file.gif` (absolute path) No Jar file is used.

4.8 Enabling Language Detection

Oracle Forms architecture supports deployment in multiple languages. The purpose of this feature is to automatically select the appropriate configuration to match a user's preferred language. In this way, all users can run Oracle Forms applications using the same URL, yet have the application run in their preferred language. As Oracle Forms

Services do not provide an integrated translation tool, you must have translated application source files.

4.8.1 Specifying Language Detection

For each configuration section in the **Web Configuration** page, you can create language-specific sections with names like `<config_name>.<language-code>`. For example, if you created a configuration section "hr", and wanted to create French and Chinese languages, your configuration section might look like the following:

```
[hr]
lookAndFeel=oracle
width=600
height=500
envFile=default.env
workingDirectory=/private/apps/hr
[hr.fr]
envFile=french.env
workingDirectory=/private/apps/hr/french
[hr.zh]
envFile=chinese.env
workingDirectory=/private/apps/hr/chinese
```

4.8.2 Inline IME Support

Inline IME support enables Forms Web applications to properly display the composing text in which each character may not be directly represented by a single keystroke (for example, Asian characters) near the insertion cursor (so called inline, or on-the-spot). It is enabled by default. To disable, set the applet parameter "inlineIME" to "false" in the baseHTML file:

```
<HTML>
<!-- FILE: base.htm (Oracle Forms) -->
 <BODY>
 ...
 <OBJECT classid=...
>
<PARAM NAME="inlineIME" VALUE="false">
<EMBED SRC="" ...
inlineIME="false"
>
...
.
</BODY>
</HTML>
```

For more information about using baseHTML, see Appendix C.4, "base.htm and basejpi.htm Files".

4.8.3 How Language Detection Works

When the Forms servlet receives a request for a particular configuration (for example, `http://myserv/servlet/frmservlet?config=hr`) it gets the client language setting from the request header `"accept-language"`. This gives a list of languages in order of preference. For example, accept-language: de, fr, en_us means the order of

preference is German, French, then US English. The servlet looks for a language-specific configuration section matching the first language. If one is not found, it looks for the next and so on. If no language-specific configuration is found, it uses the base configuration.

When the Forms servlet receives a request with no particular configuration specified (with no `"config="` URL parameter, for example, `http://myserv/servlet/frmservlet`), it looks for a language-specific section in the default section matching the first language (for example, [.fr]).

4.8.3.1 Multi-Level Inheritance

For ease of use, to avoid duplication of common values across all language-specific variants of a given base configuration, only parameters which are language-specific to be defined in the language-specific sections are allowed. Four levels of inheritance are now supported:

1. If a particular configuration is requested, using a URL query parameter like `config=myconfig`, the value for each parameter is looked for in the langage-specific configuration section which best matches the user's browser language settings (for example in section [myconfig.fr]),
2. Then, if not found, the value is looked for in the base configuration section ([myconfig],
3. Then, failing that, in the language-specific default section (for example, [.fr]),
4. And finally in the default section.

Typically, the parameters which are most likely to vary from one language to another are `"workingDirectory"` and `"envFile"`. Using a different `envFile` setting for each language lets you have different values of `NLS_LANG` (to allow for different character sets, date and number formats) and `FORMS_PATH` (to pick up language-specific fmx files). Using different `workingDirectory` settings provides another way to pick up language-specific .fmx files.

4.9 Enabling Key Mappings

A key binding connects a key to an application function. When you bind a key to a function, the program performs that function when you type that keystroke. You define key bindings in the `fmrweb.res` file in the `$ORACLE_INSTANCE/config/FormsComponent/forms/admin/resource/<lang>` directory in UNIX, for example `$ORACLE_INSTANCE/config/FormsComponent/forms/admin/resource/US`. For Windows, the location is `ORACLE_INSTANCE\config\FormsComponent\forms`.

By defining key bindings, you can integrate a variety of keyboards to make an application feel similar on each of them. On some platforms not all keys are able to be re-mapped. For example, on Microsoft Windows, because keys are defined in the Windows keyboard device driver, certain keys cannot be re-mapped. Key combinations integral to Windows, such as Alt-F4 (Close Window) and F1 (Help) cannot be re-mapped. As a general rule, keys which are part of the "extended" keyboard also cannot be re-mapped. These keys include the number pad, gray arrow and editing keys, Print Screen, Scroll Lock, and Pause.

> **Note:** If running with different NLS_LANG settings, for example, NLS_LANG=GERMAN_GERMANY=WE8ISO8859P1, a different resource file, fmrwebd.res, is used. There is a resource file for each supported language. To override this, pass parameter term=fullpath\filename.res to the Oracle Forms Runtime process.

It is possible to pass this parameter directly within the URL. For example:

```
http://hostname:port/forms/frmservlet?Form=test.fmx&term=fullpath/filename.res
```

You can also set this parameter in the formsweb.cfg file, for example:

```
otherParams=term=fullpath\filename.res
```

4.9.1 Customizing fmrweb.res

fmrweb.res is a text file which can edited with a text editor such as vi in UNIX or Notepad or Wordpad on Windows. Unlike Oracle *6i* Forms, Oracle Terminal editor is no longer required. The text file is self-documented.

> **Note:** The customization is limited, particularly compared to character mode forms. You *cannot* edit fmrweb.res with Oracle Enterprise Manager Fusion Middleware Control.

4.9.1.1 Example change: Swapping Enter and Execute Mappings

In the section marked USER-READABLE STRINGS, find the entries with

```
122 : 0 : "F11" : 76 : "Enter Query"
122 : 2 : "Ctrl+F11" : 77 : "Execute Query"
```

and change them to:

```
122 : 2 : "Ctrl+F11" : 76 : "Enter Query"
122 : 0 : "F11" : 77 : "Execute Query"
```

> **Note:** By default fmrweb.res does *not* reflect the Microsoft Windows client/server keyboard mappings. It reflects the key mapping if running client/server on UNIX X-Windows/Motif.

A file called fmrpcweb.res has also been provided which gives the Microsoft Windows client/server keyboard mappings. To use this file, rename fmrpcweb.res to fmrweb_orig.res, and copy fmrpcweb.res to fmrweb.res. Alternatively, use the term parameter as described above.

4.9.1.2 Exceptions/ Special Key Mappings

The following examples show special key mappings:

- Section 4.9.1.2.1, "Mapping F2"
- Section 4.9.1.2.2, "Mapping for ENTER to Fire KEY-ENTER-TRIGGER"
- Section 4.9.1.2.3, "Mapping Number Keys"
- Section 4.9.1.2.4, "Mapping for ESC Key to exit out of a Web Form"

4.9.1.2.1 Mapping F2

To map F2, change the default entry for F2, "List Tab Pages", to another key. Here is an example of the default entry:

```
113: 0 : "F2" : 95 : "List Tab Pages"
```

This must be explicitly changed to another key mapping such as the following:

```
113: 8 : "F2" : 95 : "List Tab Pages"
```

To map the F2 function to the F2 key, comment out the lines that begin with "113 : 0" and "113 : 8" with a # symbol and add the following lines to the bottom of the resource file:

```
113: 0 : "F2" : 84 : "Function 2"
113: 8 : " " : 95 : " "
```

Since a new function has been added which uses F2 by default, it is necessary to explicitly map this new function to something else to map the F2 key. This function was added to allow for keyboard navigation between the tab canvas pages and it defaults to F2. Even if it is commented out and not assigned to F2, the F2 key cannot be mapped unless this function, Forms Function Number 95, is mapped to another key.

4.9.1.2.2 Mapping for ENTER to Fire KEY-ENTER-TRIGGER

By default, whether deploying client/server or over the Web pressing the ENTER key takes the cursor to the next navigable item in the block. To override this default behavior it is necessary to modify the forms resource file to revise the key mapping details.

Modify `fmrweb.res` and change the Forms Function Number (FFN) from 27 to 75 for the Return Key. The line should be changed to the following:

```
10 : 0 : "Return" : 75 : "Return"
```

By default, the line is displayed with an FFN of 27 and looks as follows:

```
10 : 0 : "Return" : 27 : "Return"
```

This line should NOT fire the Key-Enter trigger since the Return or Enter key is actually returning the Return function represented by the FFN of 27. The FFN of 75 represents the Enter function and fires the Key-Enter trigger.

4.9.1.2.3 Mapping Number Keys

The objective is to map CTRL+<number> keys in `fmrweb.res` for numbers 0 to 9 and there are no Java Function keys mentioned for the numbers in fmrweb.res. The steps to be performed along with an example that shows the steps needed to map CTRL+1 to 'Next Record' are:

1. List the Java function key numbers that could be implemented in `fmrweb.res` file for the Key Mapping. For example:

   ```
   public static final int VK_1 = 0x31;
   ```

2. The hexadecimal values have to be converted to their decimal equivalents before their use in `fmrweb.res`.

   In step (1), 0x31 is a hexadecimal value that has to be converted to its decimal equivalent. (Note:1019580.6). For example,

   ```
   SQL> select hextodec('31') from dual;
   HEXTODEC('31')
   --------------
   ```

```
49
```

3. Use this decimal value for mapping the number key 1 in `fmrweb.res`. For example, CTRL+1 can be mapped to 'Next Record' as:

```
49 : 2 : "CTRL+1" : 67 : "Next Record"
```

4.9.1.2.4 Mapping for ESC Key to exit out of a Web Form

1. Make a backup copy of `fmrweb.res`.
2. Open the `fmrweb.res` file present in the path `ORACLE_HOME/FORMS` and add the following entry in it:

```
27 : 0 : "Esc" : 32 : "Exit"
```

3. Ensure that you comment or delete the old entry

```
#115 : 0 : "F4" : 32 : "Exit"
```

The first number (115) might differ on different versions or platforms. When you run the Web Form and press the ESC key, then the Form exits.

5

Using Oracle Forms Services with the HTTP Listener and Oracle WebLogic Server

Oracle WebLogic Server is a scalable, enterprise-ready Java EE application server. It implements the full range of Java EE technologies, and provides many more additional features such as advanced management, clustering, and Web services. It forms the core of the Oracle Fusion Middleware platform, and provides a stable framework for building scalable, highly available, and secure applications.

This chapter contains the following sections:

5.1 About the Oracle WebLogic Managed Server

Managed Servers host business applications, application components, Web services, and their associated resources. To optimize performance, managed servers maintain a read-only copy of the domain's configuration document. When a managed server starts up, it connects to the domain's administration server to synchronize its configuration document with the document that the administration server maintains.

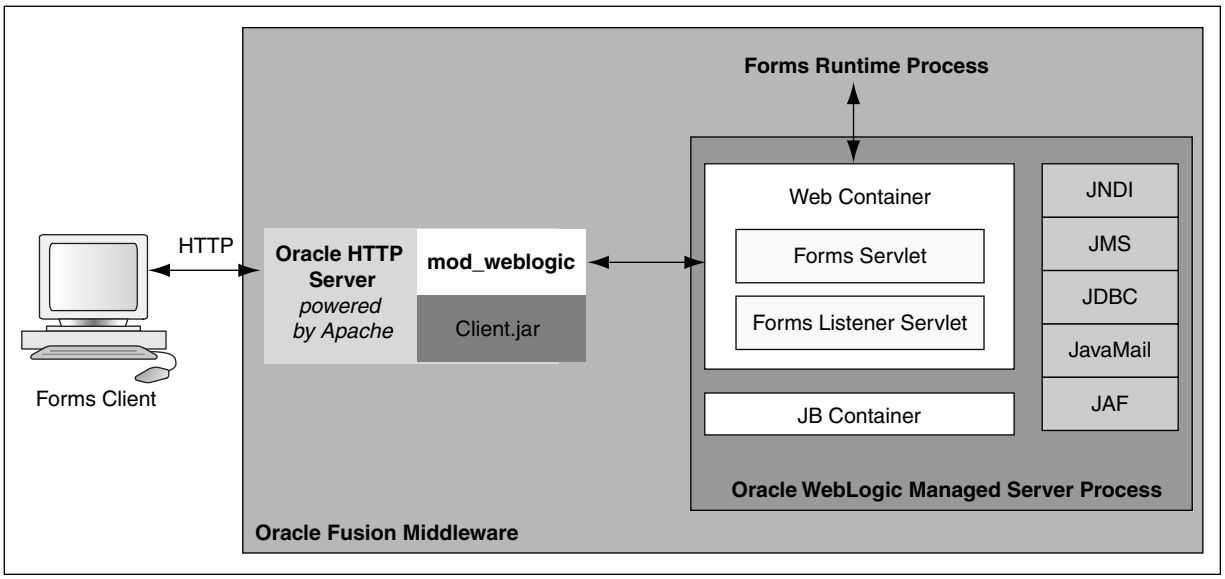
Oracle Fusion Middleware system components (such as SOA, WebCenter Portal, and Identity Management components), as well as customer-deployed applications, are deployed to managed servers in the domain.

During configuration, some managed servers are created specifically to host the Oracle Fusion Middleware system components (for example, wls_soa, wls_portal, and wls_forms).

Figure 5–1 shows a simple scenario of the Oracle WebLogic Managed Server. In the left side of the image, the Forms servlet renders the start HTML file and provides the information about the Forms Listener servlet to the client. An HTTP request is then received by the Oracle HTTP Server Listener, which passes it off to the Forms Listener servlet running inside Oracle WebLogic Managed Server, in the right side of the

image. The Forms Listener servlet establishes a runtime process and is responsible for on-going communication between the client browser and the runtime process. As more users request Oracle Forms sessions, the requests are received by the Oracle HTTP Server Listener. The HTTP Listener again passes them off to the Forms Listener servlet, which establishes more runtime processes. The Forms Listener servlet can handle many Forms runtime sessions simultaneously. While there is, of course, a limit to the number of concurrent users, the architecture presents a number of opportunities for tuning and configuration to achieve better performance (see the next section).

Figure 5–1 Oracle WebLogic Managed Server and Forms Services



5.2 Working with Forms Managed Server

By default (out-of-the-box installation), the Forms Services Java EE application (`formsapp.ear`) is deployed on Forms Managed Server (`WLS_FORMS`). You can manage `WLS_FORMS` and `formsapp.ear` using Oracle WebLogic Administration Console or Oracle Enterprise Manager Fusion Middleware Control. Refer to the following topics for more information:

- Starting and Stopping Forms Managed Server: For more information, refer to "Overview of Starting and Stopping Procedures" in *Oracle Fusion Middleware Administrator's Guide*.
- Deploying Forms Application to Forms Managed Server: For more information, refer to "Install an Enterprise application" in WebLogic Administration Console Online Help. For information on deploying, undeploying, and redeploying applications, see "Deploying Applications" in *Oracle Fusion Middleware Administrator's Guide*.
- Custom deployment of Forms Java EE application: For more information, refer to Section 5.2.1, "Custom Deployment of Forms Java EE Application".
- Expanding Forms Managed Server Clusters: For more information, refer to Section 5.2.2, "Expanding Forms Managed Server Clusters".
- Managing Cloned Managed Servers: For more information on using Fusion Middleware Control to manage cloned managed servers, see Section 5.2.3, "Registering Forms Java EE Applications."

- Modifying `weblogic.xml`, `web.xml`, `application.xml` and `weblogic-application.xml` post deployment: For more information, refer to Section 5.2.4, "Modification of Forms J2EE Application Deployment Descriptors".
- Starting Forms Managed Server as a Windows Service: For more information, refer to "Setting Up a WebLogic Server Instance as a Windows Service" in *Oracle Fusion Middleware Managing Server Startup and Shutdown for Oracle WebLogic Server*.

5.2.1 Custom Deployment of Forms Java EE Application

To create a custom managed server and deploy Forms application on it, perform the following steps:

5.2.1.1 Prerequisite Steps

1. Set the following environment variables to the paths specified:
   - `MW_HOME`: Set this variable to point to the Oracle Middleware Home location (for more information, see "A.9 Specify Installation Location Screen" in *Oracle Fusion Middleware Installation Guide for Oracle Portal, Forms, Reports and Discoverer*).
   - `ORACLE_HOME`: Set this variable with the absolute path of the Oracle Home directory. For more information, see "A.9 Specify Installation Location Screen" in *Oracle Fusion Middleware Installation Guide for Oracle Portal, Forms, Reports and Discoverer*.
   - `DOMAIN_HOME`: Set this variable with the location of the folder created by Oracle WebLogic Server for the domain specified in "A.7 Select Domain Screen" in *Oracle Fusion Middleware Installation Guide for Oracle Portal, Forms, Reports and Discoverer.*
2. Specify the JDK path in the system path.

   Enter the path to the Java executable. For example on UNIX operating systems, enter`$MW_HOME/jdk<version>/bin` in the system path (on Windows operating systems, the path is `%MW_HOME%\jdk<version>\bin`).
3. Create a managed server, for example, `WLS_FORMS_CUSTOM_APP`, as part of the same cluster as the default managed server (`WLS_FORMS`).

   For more information on adding a managed server, refer to "Adding Additional Managed Servers to a Domain" in *Oracle Fusion Middleware Administrator's Guide*.
4. Specify the following properties of the managed server using the WebLogic Administration Console.
   - Classpath: Specify the value: `<ORACLE_HOME>/opmn/lib/optic.jar` (on Windows operating systems: `<ORACLE_HOME>\opmn\lib\optic.jar`). Replace `<ORACLE_HOME>` with the absolute path.
   - Arguments: Specify the following values:

     ```
     -Dclassic.oracle.home=<ORACLE_HOME> -Doracle.instance=<ORACLE_INSTANCE> -
     Doracle.instance.name=<ORACLE_INSTANCE_NAME> -Doracle.forms.weblogic=1
     ```

     Make sure all the entries are in a single line (without any carriage returns). Replace *<ORACLE_HOME>*, *<ORACLE_INSTANCE>* with the absolute paths. Replace *<ORACLE_INSTANCE_NAME>* with the name of the Oracle Instance (default name asinst_1).

For more information, refer to "Server Start" in *Oracle WebLogic Administration Console Help*.

5. Perform the following steps to create a folder structure in `ORACLE_HOME`:

   a. On UNIX operating systems, create a new folder for the custom application.

      For example, create `customapp` as follows:

      ```
      mkdir -p $ORACLE_HOME/customapp
      ```

      Create a Java folder in customapp and create a symbolic link for the folder as follows:

      For example:

      ```
      cd $ORACLE_HOME/customapp
      ln -s $ORACLE_HOME/forms/java $ORACLE_HOME/customapp/java
      ```

      Copy the application files to the new folder.

      For example:

      ```
      cp -rpf $ORACLE_HOME/forms/j2ee $ORACLE_HOME/customapp/
      ```

   b. On Windows operating systems, use the following commands to create a folder structure under ORACLE_HOME directory:

      ```
      mkdir %ORACLE_HOME%\customapp\java
      mkdir %ORACLE_HOME%\customapp\j2ee
      cd %ORACLE_HOME%\customapp
      xcopy /S /E %ORACLE_HOME%\forms\java %ORACLE_
      HOME%\customapp\java
      xcopy /S /E %ORACLE_HOME%\forms\j2ee %ORACLE_
      HOME%\customapp\j2ee
      ```

5.2.1.2 Override the Default Servlet Alias and the Context Root

1. Extract the EAR file.

   For example, on UNIX operating systems:

   ```
   cd $ORACLE_HOME/customapp/j2ee
   jar xvf formsapp.ear
   ```

   On Windows operating systems:

   ```
   cd %ORACLE_HOME%\customapp\j2ee
   jar xvf formsapp.ear
   ```

2. Extract the WAR file.

   For example, on UNIX operating systems:

   ```
   mkdir -p $ORACLE_HOME/customapp/j2ee/warfile
   cd $ORACLE_HOME/customapp/j2ee/warfile
   jar xvf $ORACLE_HOME/customapp/j2ee/formsweb.war
   ```

   On Windows operating systems:

   ```
   mkdir %ORACLE_HOME%\customapp\j2ee\warfile
   ```

```
cd %ORACLE_HOME%\customapp\j2ee\warfile

jar xvf %ORACLE_HOME%\customapp\j2ee\formsweb.war
```

3. Override the servlet alias in web.xml deployment descriptor that is located in the WEB-INF folder.

   For example, on UNIX operating systems:

   ```
   cd $ORACLE_HOME/customapp/j2ee/warfile/WEB-INF
   ```

   On Windows operating systems:

   ```
   cd %ORACLE_HOME%\customapp\j2ee\warfile\WEB-INF
   ```

   Edit web.xml in an editor and replace frmservlet with customservlet (entries under tags <Servlet-Name>, <url-pattern>, <welcome-file>).

4. Repackage the WAR file.

   For example, on UNIX operating systems:

   ```
   cd $ORACLE_HOME/customapp/j2ee/warfile

   jar cvfM formsweb.war ./*

   mv formsweb.war $ORACLE_HOME/customapp/j2ee/
   ```

   On Windows operating systems:

   ```
   cd %ORACLE_HOME%\customapp\j2ee\warfile

   jar cvfM formsweb.war .\*

   copy formsweb.war %ORACLE_HOME%\customapp\j2ee\

   del formsweb.war
   ```

5. Override the application context root in application.xml deployment descriptor that is located in the META-INF folder.

   For example, on UNIX operating systems:

   ```
   cd $ORACLE_HOME/customapp/j2ee/META-INF
   ```

   On Windows operating systems:

   ```
   cd %ORACLE_HOME%\customapp\j2ee\META-INF
   ```

   Edit application.xml, change context-root to customapp.

6. Modify the codebase and serverURL entries in formsweb.cfg.

   For example, on UNIX operating systems:

   ```
   cd $ORACLE_HOME/customapp/j2ee/config
   ```

   On Windows operating systems:

   ```
   cd %ORACLE_HOME%\customapp\j2ee\config
   ```

   Edit `formsweb.cfg` and change the context-root entries in serverURL and codebase parameters.

   For example,

   Change `serverURL=/forms/lservlet` to `serverURL=/customapp/lservlet`.

   Change codebase from `/forms/java` to `customapp/java`.

7. Repackage the EAR file.

For example, on UNIX operating systems:

```
cd $ORACLE_HOME/customapp/j2ee
```

```
jar cvfM customapp.ear META-INF/MANIFEST.MF APP-INF/*
config/* formsweb.war META-INF/*
```

On Windows operating systems:

```
cd %ORACLE_HOME%\customapp\j2ee
```

```
jar cvfM customapp.ear META-INF\MANIFEST.MF APP-INF\*
config\* formsweb.war META-INF\*
```

8. Clean the extracted EAR file contents. On UNIX operating systems:

```
rm -rf META-INF APP-INF config META-INF formsweb.war
```

On Windows operating systems:

```
RMDIR META-INF APP-INF config META-INF /s /q
```

```
DEL formsweb.war
```

5.2.1.3 Create the Deployment Plan

1. Create a folder in customapp named 11.1.1.

For example, on UNIX operating systems:

```
mkdir -p $DOMAIN_HOME/deploymentplans/customapp/11.1.1
```

On Windows operating systems,

```
mkdir %DOMAIN_HOME%\deploymentplans\customapp\11.1.1
```

2. Copy the following entries to a file `$DOMAIN_HOME/deploymentplans/customapp/11.1.1/plan.xml` (on Windows operating systems, `%DOMAIN_HOME%\deploymentplans\customapp\11.1.1\plan.xml`). Example 5–1 describes a deployment plan with application name of `customapp` and managed server name of `WLS_FORMS_CUSTOM_APP`. Ensure you make the following changes:

   - Replace the custom application name, location of EAR file, and managed server with the names and locations in your environment.
   - Replace *<DOMAIN_HOME>*, *<ORACLE_HOME>* with the absolute paths.

Example 5–1 Example of Deployment Plan

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.oracle.com/weblogic/deployment-plan
 http://xmlns.oracle.com/weblogic/deployment-plan/1.0/deployment-plan.xsd"
 global-variables="false">
  <application-name>customapp</application-name>
  <variable-definition>
    <variable>
      <name>vd-<ORACLE_HOME>/customapp</name>
      <value><ORACLE_HOME>/customapp</value>
    </variable>
    <variable>
      <name>vd-<DOMAIN_HOME>/config/fmwconfig/servers/WLS_FORMS_CUSTOM_
APP/applications/customapp_11.1.1/config/customapp</name>
```

```
      <value><DOMAIN_HOME>/config/fmwconfig/servers/WLS_FORMS_CUSTOM_
APP/applications/customapp_11.1.1/config/customapp</value>
    </variable>
  </variable-definition>
  <module-override>
    <module-name>customapp.ear</module-name>
    <module-type>ear</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-application</root-element>
      <uri>META-INF/weblogic-application.xml</uri>
    </module-descriptor>
    <module-descriptor external="false">
      <root-element>application</root-element>
      <uri>META-INF/application.xml</uri>
    </module-descriptor>
    <module-descriptor external="true">
      <root-element>wldf-resource</root-element>
      <uri>META-INF/weblogic-diagnostics.xml</uri>
    </module-descriptor>
  </module-override>
  <module-override>
    <module-name>formsweb.war</module-name>
    <module-type>war</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-web-app</root-element>
      <uri>WEB-INF/weblogic.xml</uri>
      <variable-assignment>
        <name>vd-<ORACLE_HOME>/customapp</name>

<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="java/*"]/l
ocal-path</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>vd-<ORACLE_HOME>/customapp</name>

<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="webutil/*"
]/local-path</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>vd-<DOMAIN_HOME>/config/fmwconfig/servers/WLS_FORMS_CUSTOM_
APP/applications/customapp_11.1.1/config/customapp</name>

<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="registry/*
"]/local-path</xpath>
      </variable-assignment>
    </module-descriptor>
    <module-descriptor external="false">
      <root-element>web-app</root-element>
      <uri>WEB-INF/web.xml</uri>
    </module-descriptor>
  </module-override>
</deployment-plan>
```

5.2.1.4 Deploy the Custom EAR file

Deploy the custom EAR file using WebLogic Scripting Tool (WLST) commands. For example, on UNIX operating systems:

```
$MW_HOME/oracle_common/common/bin/wlst.sh
```

On Windows operating systems: `%MW_HOME%\oracle_common\common\bin\wlst.cmd`

Use the WLST `deploy` command to deploy the application:

```
wls:/offline> connect('weblogic','welcome1')

wls:/ClassicDomain/serverConfig> deploy('customapp', '<ORACLE_HOME>/customapp/j2ee/customapp.ear',  'WLS_FORMS_CUSTOM_APP', 'nostage','<DOMAIN_HOME>/deploymentplans/customapp/11.1.1/plan.xml')
```

Be sure to make the following changes in the command:

- Replace customapp with actual context root.
- Replace *<DOMAIN_HOME>*, *<ORACLE_HOME>* with the absolute paths.

5.2.1.5 Post-Patching Tasks

If you have patched your existing Oracle Fusion Middleware 11*g* Patch Set 1 (Release 11.1.1.2.0) or Patch Set 2 (Release 11.1.1.3.0) environment to consume the latest patch set, perform the following steps:

1. Copy the Forms J2EE application files to the customapp directory. On Unix operating systems:

   ```
   cp -rpf $ORACLE_HOME/forms/j2ee/* $ORACLE_HOME/customapp/j2ee/*
   ```

   On Windows operating systems:

   ```
   xcopy /S /E %ORACLE_HOME%\forms\java %ORACLE_HOME%\customapp\java

   xcopy /S /E %ORACLE_HOME%\forms\j2ee %ORACLE_HOME%\customapp\j2ee
   ```

2. Repeat the steps in "Override the Default Servlet Alias and the Context Root".
3. Restart the custom managed server.

5.2.1.6 Test the Custom Deployment

Test the deployment using the URL: `http://<Host>:<Port Number>/<context root>/<servlet name>`.

For the example in this section, the URL would be `http://<Host>:<Port Number>/customapp/customservlet`.

5.2.2 Expanding Forms Managed Server Clusters

To improve the scalability and performance of Forms deployments on high-end machines (multiprocessor and high-memory configuration machines), expand the Forms Managed Server cluster (`cluster_forms`). Perform the following manual steps to expand the Forms Managed Server cluster:

1. Perform the following steps to add a new Managed Server to the cluster (`cluster_forms`):

   a. Using the Oracle WebLogic Server Administration Console, you can choose to either clone the default Forms Managed Server (`WLS_FORMS`) or create a new Managed Server (for example, `WLS_FORMS_1`, with port number 9010).

For more information on using Fusion Middleware Control to manage the new or cloned managed server, see Section 5.2.3, "Registering Forms Java EE Applications".

b. In the Server Properties page, add the newly created Managed Server to the Forms cluster `cluster_forms`.

c. In the General Tab, assign a port number to the Managed Server.

d. Assign a machine to the Managed Server.

2. Perform the following steps to edit the configuration of the new managed server:

a. Using the Oracle WebLogic Server Administration Console, in the Server Start Tab, set the following Server Start properties.

b. Add the following system properties without any carriage returns to the arguments:

```
-Dclassic.oracle.home=<ORACLE_HOME
location>-Doracle.instance=<ORACLE_INSTANCE
location>-Doracle.instance.name=<ORACLE_INSTANCE Name>

-Doracle.forms.weblogic=1
```

c. Add the following to the CLASSPATH: `<ORACLE_HOME>/opmn/lib/optic.jar:<FMW_HOME>/oracle_common/modules/oracle.ldap_11.1.1/ldapjclnt11.jar:<FMW_HOME>/oracle_common/jlib/rcucommon.jar`

3. Activate the changes and start the new Managed Server.

4. Add the new Managed Server's host and port information to the WebLogicCluster entry in `forms.conf`.

```
<Location /forms>

   SetHandler weblogic-handler

   WebLogicCluster <HostName>:9001, <HostName>:9010

   DynamicServerList OFF

</Location>
```

5. Restart OHS.

5.2.3 Registering Forms Java EE Applications

To use Fusion Middleware Control to manage the new or cloned managed servers under the default Forms WLS cluster, you register the Forms Java EE applications.

Perform the following steps to register the Forms Java EE applications:

1. Create a sample WLST script as shown in Example 5–2. In this example, the script is named formsappRegistration.py.

Example 5–2 Sample WLST Script

```
#
# formsappRegistration.py
# Workaround script to register/unregister Forms J2EE application Mbean
# as a member of Forms System Component Mbean
#
from javax.management import ObjectName, Attribute
from jarray import array
```

```
import getopt, sys
#
# function prints the usage
#
def usage():
     message =
"-------------------------------------------------------------------------------"
 + "Usage : " + " $FMW_HOME/oracle_common/common/bin/wlst.sh " + sys.argv[0] + "
 --adminServerName=<admin server name> --asinstName=<Oracle Instance name>
 --managedServer=<newly added Forms managed server name>  --formsappName=<forms
 J2EE application name> -o <option> " + "  valid option - registerApp or
 unregisterApp" + " examples:" + "  $FMW_HOME/oracle_common/common/bin/wlst.sh " +
 sys.argv[0] +" --adminServerName=AdminServer --asinstName=asinst_1
 --managedServer=WLS_FORMS1 --formsappName=formsapp -o registerApp " + " $FMW_
HOME/oracle_common/common/bin/wlst.sh " + sys.argv[0] + "
 --adminServerName=AdminServer --asinstName=asinst_1 --managedServer=WLS_FORMS1
 --formsappName=formsapp -o unregisterApp " +
"-------------------------------------------------------------------------------"
     print message
#
# getFormsCompMbeanObjectName - function to generate the Forms System Component
#                                                        Mbean ObjectName.
#
def getFormsCompMbeanObjectName(asInstName, adminServerName):
    frmCompONameString = "oracle.as.management.mbeans.register:" \
                         + "Location="+ adminServerName +
",type=SystemComponent,name=/" \
                         + asinstName +"/forms,instance=" + asinstName \
                         + ",component=forms,EMTargetType=oracle_forms";
    print frmCompONameString
    frmCompOName = ObjectName(frmCompONameString)
    return frmCompOName
#
# getFormsAppMbeanObjectName - function to generate the Forms J2EE application
#                                                       ObjectName.
#
def getFormsAppMbeanObjectName(appName, managedServer):
    frmappONameString = "com.bea:Name="+formsappName+ "#11.1.1,Location=" +\
                        managedServer+ ",Type=AppDeployment"
    frmappOName = ObjectName(frmappONameString)
    return frmappOName
#
# doesMemberExist - utility function to check if app is already registered as a
member
#
def doesMemberExist(member, list):
    for item in list:
        if item == member:
            return 1
    return None
#
#  registerFormsApp - registers Forms J2EE application Mbean as a member of
#                               Forms System Component Mbean
#
def registerFormsApp(formsCompMbean, frmappMbean):
    domainRuntime()
    membersArray = mbs.getAttribute(formsCompMbean,"Members")
    membersList = membersArray.tolist()

    if membersList == []:
```

```
        print "Members list is empty"
    else:
        print "Members list is not empty"

        if doesMemberExist(frmappMbean, membersList):
            print "Member already registered, skipping registration"
        else:
            print "Member is not found, append it to the members list"
            membersList.append(frmappMbean)
            membersArray = array(membersList, ObjectName)
            membersAttrib = Attribute("Members",membersArray)
            mbs.setAttribute(formsCompMbean, membersAttrib)
#
#  unregisterFormsApp - unregisters Forms J2EE application Mbean as a member of
#                               Forms System Component Mbean
#
def unregisterFormsApp(formsCompMbean, frmappMbean):
    domainRuntime()
    membersArray = mbs.getAttribute(formsCompMbean,"Members")
    membersList = membersArray.tolist()

    if membersList == []:
        print "Members list is empty"
    else:
        print "Members list is not empty"

        if doesMemberExist(frmappMbean, membersList):
            print "Found the Member, removing it."
            membersList.remove(frmappMbean)
            membersArray = array(membersList, ObjectName)
            membersAttrib = Attribute("Members",membersArray)
            mbs.setAttribute(formsCompMbean, membersAttrib)
        else:
            print "Member not found, skipping unregister"
#
# execution starts here
#

if len(sys.argv) != 7 :
    print "invalid arguments passed to the script"
    usage()
    sys.exit(0)

# trim the first argument which is the name of the script

args = sys.argv[1:7]
optlist, args = getopt.getopt(args,'o', [
'adminServerName=','asinstName=','managedServer=','formsappName='])
options = dict(optlist)

adminServerName = options["--adminServerName"]
asinstName = options["--asinstName"]
managedServer = options["--managedServer"]
formsappName  = options["--formsappName"]

if adminServerName == [] or \
   managedServer == [] or formsappName == [] or not args:
    print "invalid arguments passed to the script "
    usage()
    sys.exit(0)
```

```
argument = args[0]
print "enter the WLST connection paramters ..."
connect()

frmcompMbean = getFormsCompMbeanObjectName(asinstName,adminServerName)
print frmcompMbean

frmappMbean = getFormsAppMbeanObjectName(formsappName,managedServer)
print frmappMbean

if argument == "registerApp":
    print "registering Forms J2EE application " + formsappName
    registerFormsApp(frmcompMbean,frmappMbean)
elif argument == "unregisterApp":
    print "unregistering Forms J2EE application " + formsappName
    unregisterFormsApp(frmcompMbean,frmappMbean)

else:
    print "invalid option passed to the scripts ..."
    usage()

disconnect()
print "done... "
```

2. Execute the script. You can use the help argument for more information as shown in Example 5–3.

Example 5–3 Sample Script Execution

```
$FMW_HOME/oracle_common/common/bin/wlst.sh formsappRegistration.py help

-------------------------------------------------------------------------------
Usage :
 $FMW_HOME/oracle_common/common/bin/wlst.sh formsappRegistration.py
 --adminServerName=<admin server name> --asinstName=<Oracle Instance name>
 --managedServer=<newly added Forms managed server name>  --formsappName=<forms
 J2EE application name> -o <option>

 valid options - registerApp or unregisterApp

examples:
  $FMW_HOME/oracle_common/common/bin/wlst.sh formsappRegistration.py
 --adminServerName=AdminServer --asinstName=asinst_1 --managedServer=WLS_FORMS1
 --formsappName=formsapp -o registerApp

 $FMW_HOME/oracle_common/common/bin/wlst.sh formsappRegistration.py
 --adminServerName=AdminServer --asinstName=asinst_1 --managedServer=WLS_FORMS1
 --formsappName=formsapp -o unregisterApp

 $FMW_HOME/oracle_common/common/bin/wlst.sh formsappRegistration.py
--adminServerName=AdminServer --asinstName=asinst_1 --managedServer=WLS_FORMS1
--formsappName=formsapp -o unregisterApp
-------------------------------------------------------------------------------
```

3. When prompted, enter the administration server username, password, and connection information.
4. Accept the default server URL.
5. Example 5–4 shows a sample of the execution and results of server registration.

Example 5–4 Sample Execution and Results

```
$FMW_HOME/oracle_common/common/bin ( ) -> ./wlst.sh formsappRegistration.py -
adminServerName=AdminServer --asinstName=asinst_1 --managedServer=WLS_FORMS1 -
formsappName=formsapp -o registerApp

CLASSPATH=. . . . . .
. . . . . .
. . . . . .
Your environment has been set.

Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

enter the WLST connection paramters ...
Please enter your username :weblogic
Please enter your password :
Please enter your server URL [t3://localhost:7001] :
Connecting to t3://localhost:7001 with userid weblogic ...
Successfully connected to Admin Server 'AdminServer' that belongs to domain
'ClassicDomain'.

Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.

registering Forms J2EE application formsapp
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help(domainRuntime)

Members list is not empty
Member is not found, append it to the members list
Disconnected from weblogic server: AdminServer
done...
```

5.2.4 Modification of Forms J2EE Application Deployment Descriptors

Post-deployment, Forms J2EE application deployment descriptors (`weblogic.xml`, `web.xml`, `application.xml` and `weblogic-application.xml`) cannot be modified in Oracle WebLogic Server.

As a workaround, perform the following steps to customize the Forms J2EE application deployment descriptors and redeploy the application:

1. Back up the default formsapp deployment plan, `$DOMAIN_HOME/deploymentplans/formsapp/11.1.1/plan.xml`.
2. Add the deployment descriptors customizations to the Forms J2EE application's deployment plan. See the "Modifying the Deployment Plan" for an example.

> **Note:** For more information on updating the deployment plan, refer to the *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server.*

3. Using the WebLogic Administration Console, update the forms application (redeploy) and select the option **Update this application in place with new deployment plan changes**.
4. Restart the Forms J2EE application using the WebLogic Administration Console.

Modifying the Deployment Plan

In this example, the deployment plan is modified to override the Forms Servlet `testMode` parameter and set it to true. To modify the deployment plan, perform the following steps:

1. Enter the following commands:

```
mkdir -p $CLASSIC_ORACLE_HOME/forms/j2ee/backup
cd $CLASSIC_ORACLE_HOME/forms/j2ee
cp $DOMAIN_HOME/deploymentplans/formsapp/11.1.1/plan.xml backup/
vi $DOMAIN_HOME/deploymentplans/formsapp/11.1.1/plan.xml
```

2. Modify the deployment plan. The following is a sample of the deployment plan with the added entries highlighted in bold:

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/weblogic/deployment-plan
http://xmlns.oracle.com/weblogic/deployment-plan/1.0/deployment-plan.xsd"
global-variables="false">
  <application-name>formsapp</application-name>
  <variable-definition>
    <variable>
      <name>vd-/scratch/t_work/Oracle/Middleware/as_1/forms</name>
      <value>/scratch/t_work/Oracle/Middleware/as_1/forms</value>
    </variable>
    <variable>
      <name>vd-/scratch/t_work/Oracle/Middleware/user_
projects/domains/ClassicDomain/config/fmwconfig/servers/WLS_
FORMS/applications/formsapp_11.1.1/config/forms</name>
      <value>/scratch/t_work/Oracle/Middleware/user_
projects/domains/ClassicDomain/config/fmwconfig/servers/WLS_
FORMS/applications/formsapp_11.1.1/config/forms</value>
    </variable>
    <variable>
       <name>FormsServlet_InitParam_testMode</name>
       <value>true</value>
    </variable>
  </variable-definition>
  <module-override>
    <module-name>formsapp.ear</module-name>
    <module-type>ear</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-application</root-element>
      <uri>META-INF/weblogic-application.xml</uri>
    </module-descriptor>
    <module-descriptor external="false">
      <root-element>application</root-element>
      <uri>META-INF/application.xml</uri>
    </module-descriptor>
    <module-descriptor external="true">
      <root-element>wldf-resource</root-element>
      <uri>META-INF/weblogic-diagnostics.xml</uri>
    </module-descriptor>
```

```
  </module-override>
  <module-override>
    <module-name>formsweb.war</module-name>
    <module-type>war</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-web-app</root-element>
      <uri>WEB-INF/weblogic.xml</uri>
      <variable-assignment>
        <name>vd-/scratch/t_work/Oracle/Middleware/as_1/forms</name>
<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="java/*"]/local
-path</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>vd-/scratch/t_work/Oracle/Middleware/as_1/forms</name>
<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="webutil/*"]/lo
cal-path</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>vd-/scratch/t_work/Oracle/Middleware/user_
projects/domains/ClassicDomain/config/fmwconfig/servers/WLS_
FORMS/applications/formsapp_11.1.1/config/forms</name>
<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="registry/*"]/l
ocal-path</xpath>
      </variable-assignment>
    </module-descriptor>
    <module-descriptor external="false">
      <root-element>web-app</root-element>
      <uri>WEB-INF/web.xml</uri>
      <variable-assignment>
        <name>FormsServlet_InitParam_testMode</name>
<xpath>/web-app/servlet/[servlet-name="frmservlet"]/init-param/[param-name="tes
tMode"]/param-value</xpath>
      </variable-assignment>
    </module-descriptor>
  </module-override>
</deployment-plan>
```

3. Using the WebLogic Administration Console, update the Forms J2EE application deployment (formsapp (11.1.1)). For more information on redeploying Forms J2EE application, refer to *Oracle Fusion Middleware Administrator's Guide*.
4. Restart the Forms J2EE application using the WebLogic Administration Console.

5.3 Performance/Scalability Tuning

The steps for tuning the Forms Listener servlet are similar to steps for tuning any high throughput servlet application. You have to take into account resource management and user needs for optimal tuning of your particular Forms Services configuration. For more information, see *Oracle Fusion Middleware Performance Guide* available on OTN at http://www.oracle.com/technology/documentation/.

5.3.1 Limit the number of HTTPD processes

To control spawning HTTPD processes (which is memory consuming) set the KeepAlive directive in the Oracle HTTP Listener configuration file (`httpd.conf`): `KeepAlive Off`

KeepAlive specifies whether or not to allow persistent connections (more than one request per connection). If you must use `KeepAlive On`, for example, for another

application, make sure that `KeepAliveTimeout` is set to a low number for example, 15 seconds, which is the default. The KeepAlive setting is used to maintain a persistent connection between the client (Browser) and the OHS server. It does not have anything to do with the OHS to Oracle WebLogic Server connection.

5.3.2 Set the MaxClients Directive to a High value

You can let the HTTP Listener determine when to create more HTTPD processes. Therefore, set the `MaxClients` directive to a high value in the configuration file (`httpd.conf`). However, you need to consider the memory available on the system when setting this parameter.

`MaxClients=256` indicates that the listener can create up to 256 HTTPD processes to handle concurrent requests.

If your HTTP requests come in bursts, and you want to reduce the time to start the necessary HTTPD processes, you can set `MinSpareServers` and `MaxSpareServers` (in `httpd.conf`) to have an appropriate number of processes ready. However, the default values of 5 and 10 respectively are sufficient for most sites.

5.4 Load Balancing Oracle WebLogic Server

The Forms Listener servlet architecture allows you to load balance the system using any of the standard HTTP load balancing techniques available.

The Oracle HTTP Server Listener provides a load balancing mechanism that allows you to run multiple WebLogic instances on the same host as the HTTP process, on multiple, different hosts, or on any combination of hosts. The HTTP Listener then routes HTTP requests to Oracle WebLogic Managed Server instances.

The following scenarios are just a few of the possible combinations available and are intended to show you some of the possibilities. The best choice for your site will depend on many factors.

For a complete description of this feature, refer to the *Oracle Fusion Middleware Performance Guide* (available on OTN at `http://www.oracle.com/technology/documentation/index.html`).

The following images illustrate four possible deployment scenarios:

- Figure 5–2 shows the Oracle HTTP Server balancing incoming requests between multiple Oracle WebLogic Managed Servers on the same host as the Oracle HTTP Listener.
- Figure 5–3 shows the Oracle HTTP Server balancing incoming requests between multiple Oracle WebLogic Managed Servers on a different host to the Oracle HTTP Listener.
- Figure 5–4: shows the Oracle HTTP Server balancing incoming requests between multiple Oracle WebLogic Managed Servers on multiple different hosts and multiple different hosts each running an Oracle HTTP Listener.
- Figure 5–5: shows the Oracle HTTP Server balancing incoming requests between multiple Oracle WebLogic Managed Servers on a single host but with multiple different hosts each running an Oracle HTTP Listener.

Figure 5–2 Multiple Oracle WebLogic Servers on the same host as the Oracle HTTP Listener

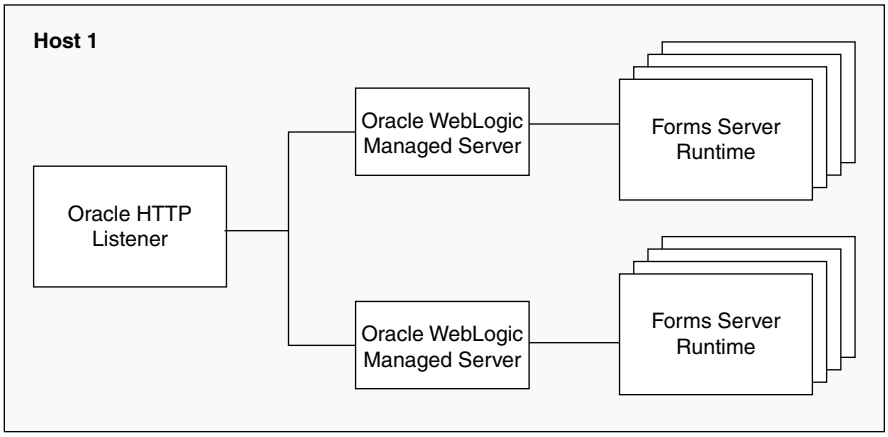


Figure 5–3 Multiple Oracle WebLogic Servers on a different host to the Oracle HTTP Listener

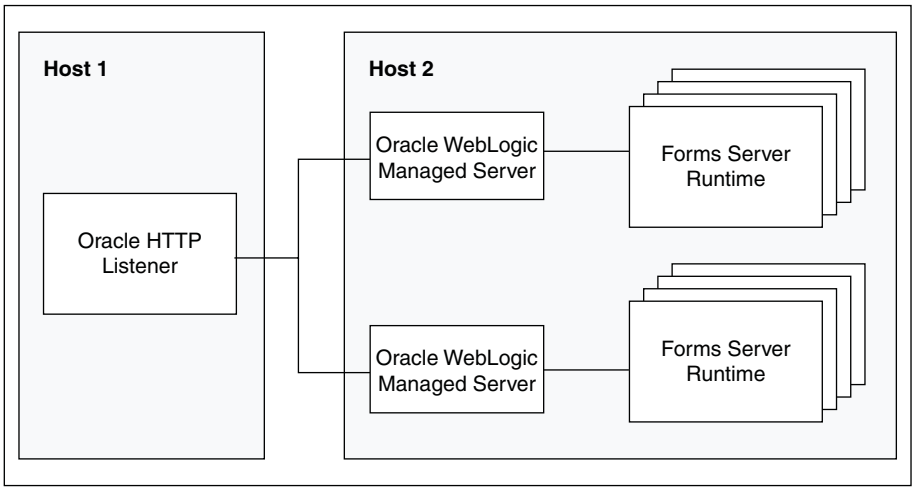


Figure 5–4 Multiple Oracle WebLogic Servers and multiple Oracle HTTP Listeners on different hosts

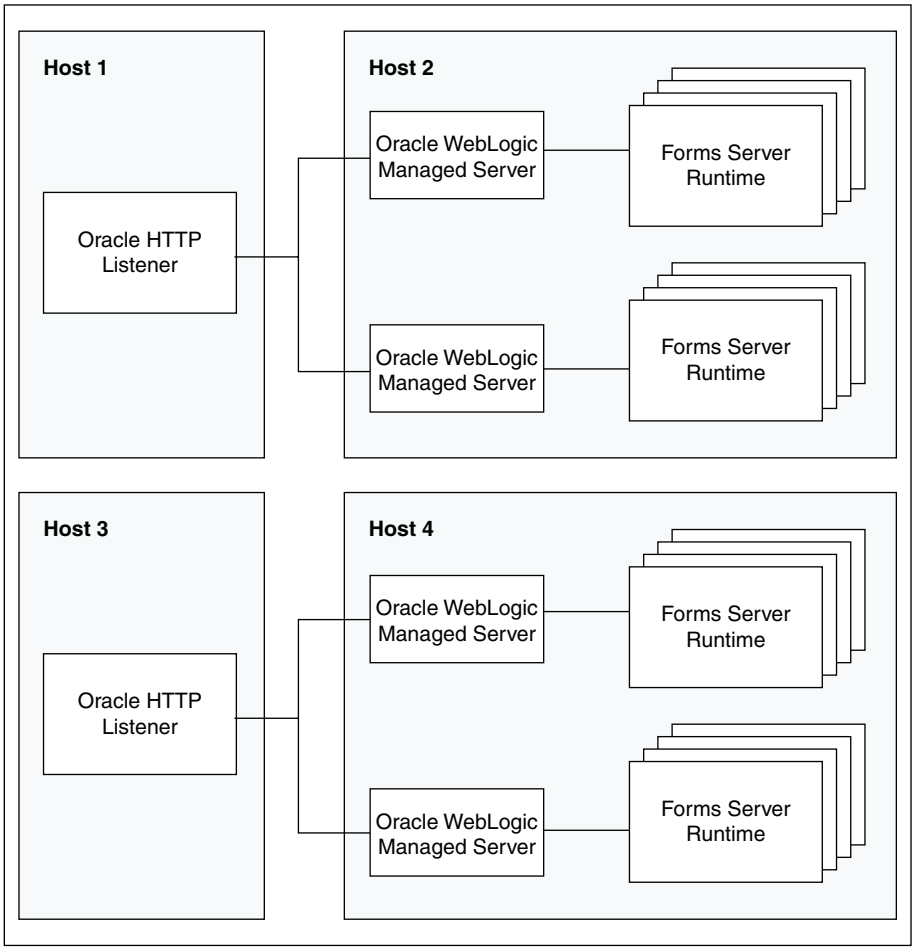
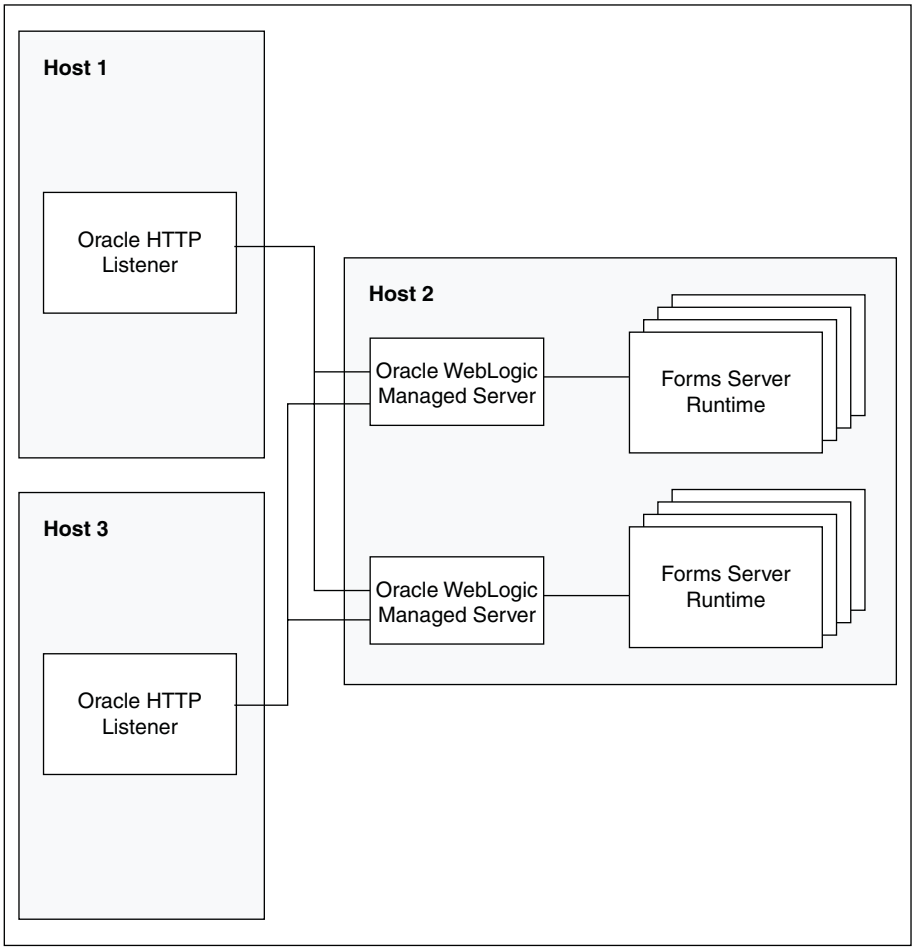


Figure 5–5 Multiple Oracle HTTP Listeners on different hosts with multiple Oracle WebLogic Servers on one host



For more information about tuning and optimizing Forms Services with the HTTP Listener and Oracle WebLogic Server, see *Oracle Fusion Middleware Performance Guide*, available on Oracle Technology Network (OTN) at http://www.oracle.com/technology/documentation/index.html.

5.5 Using HTTPS with the Forms Listener Servlet

Using HTTPS with Oracle Forms is no different than using HTTPS with any other Web-based application. HTTPS requires the use of digital certificates (for example, VeriSign). Because Forms Services servlets are accessed via your Web server, you do not need to purchase special certificates for communications between the Oracle Forms client and the server. You only need to purchase a certificate for your Web server from a recognized certificate authority.

5.6 Using an Authenticating Proxy to Run Oracle Forms Applications

The default configuration as set up by the Oracle Fusion Middleware installation process supports authenticating proxies. An authenticating proxy is one that requires the user to supply a username and password in order to access the destination server where the application is running. Typically, authenticating proxies set a cookie to

detect whether the user has logged on (or been authenticated). The cookie is sent in all subsequent network requests to avoid further logon prompts.

The codebase and server URL values that are set up by the Oracle WebLogic Server installation process include `$ORACLE_HOME/forms/java` and `/forms/lservlet`. As these are under the document base of the page (`$ORACLE_HOME/forms`), authenticating proxies will work.

5.7 Oracle Forms Services and SSL

To run Oracle Forms Services applications in SSL mode:

- Create a Wallet to manage certificates.
- Enable the HTTPS port in Oracle HTTP Server. By default, Oracle HTTP Server has one SSL Port enabled (8890).
- Enable Web Cache to accept HTTPS connections from Oracle HTTP Server.

For more information on the above topics, see the section "SSL Configuration in Oracle Fusion Middleware" in the *Oracle Fusion Middleware Administrator's Guide*.

Note: When you change the Oracle Web Cache port using Enterprise Manager, regenerate the `osso.conf` and copy the generated `osso.conf` file to `$ORACLE_INSTANCE/config/OHS/<OHS_INSTANCE>/moduleconf` directory. Restart the Oracle HTTP Server and Oracle Web Cache for the changes to take effect.

5.8 Enabling SSL with a Load Balancing Router

Running a Forms application that uses an HTTPS port requires a certificate to be imported. If Oracle Forms is behind a load balancing router, and SSL terminates at it, you need to import the certificate from the load balancing router.

To enable SSL with your Forms applications over a load balancing router:

1. Start a Web browser and enter the Forms application HTTPS URL containing the fully qualified host name (including port number if required) used by your own Oracle installation. For example:
   `https://example.com:443/forms/frmservlet`

   The Security Alert dialog box is displayed.
2. Click **View Certificate**.
3. Click the **Details** tab in the Certificate dialog.
4. Click **Copy to File...**
5. In the Welcome page of the Certificate Export Wizard, click **Next**.
6. In the Export File Format page, select **Base-64 encoded X.509 (.CER)**, then click Next.
7. Enter a file name such as `c:\temp\forms`, then click Next.
8. Click **Finish**.

   A message appears saying that the export was successful.
9. Click **OK**.

10. Close the Certificate Export Wizard, but keep the Security Alert dialog open.
11. Import the security certificate file that you saved earlier into the certificate store of the JVM you are using. For more information, see the next section.
12. At the Security Alert dialog, click Yes to accept the security certificate and start the Forms application.

Importing the certificate into Java Plugin

1. On the client machine, open the Control Panel.
2. Open Java.
3. Navigate to Securities tab.
4. Click Certificate.
5. Import the certificate that was exported in the previous section.
6. Click Apply.

6

Oracle Forms and JavaScript Integration

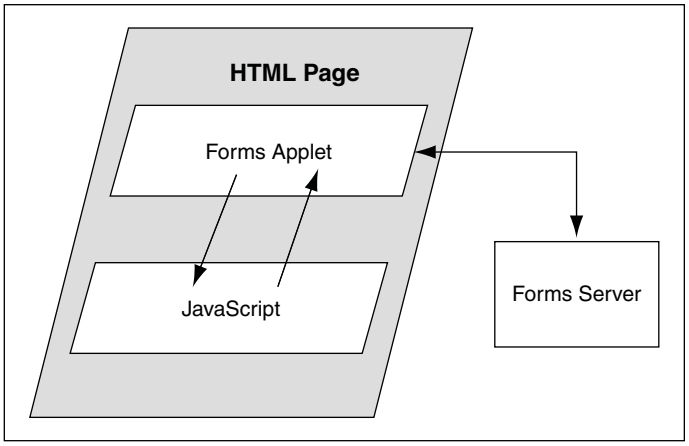
This chapter contains the following sections:

6.1 About Oracle Forms Calling External Events

In previous releases of Oracle Forms, you had to implement OLE and DDE to interact with a limited number of event types outside of Forms. In later versions, Forms offered `web.show_document` and Java integration to interface with external application sources. But in terms of calling out to the Web page where Forms is displayed, there was no easy solution. It was also not possible to call from the Web page into Forms, perhaps to update a value acquired from an HTML form.

In Oracle Forms 11*g*, JavaScript integration provides the ability to have JavaScript events call into Forms, or have Forms execute JavaScript events. Figure 6–1 shows how JavaScript and Oracle Forms work together. In the left side of the image, JavaScript is executed in the page in which the Forms applet is hosted. Oracle Forms now has the capability to call JavaScript functions using native built-ins. Also, JavaScript functions can now trigger a Oracle Forms trigger by using a new API that has been provided.

Figure 6–1 Oracle Forms and JavaScript



Two new calls are available in the web Built-in package:

- web.javascript_eval_expr
- web.javascript_eval_function

The first call web.javascript_eval_expr is a procedure which takes two arguments: an expression and a target, both of data type varchar2. This legal JavaScript expression is interpreted in the Web page in which the Forms applet is embedded. The expression can be a call to a function that is defined in the target page or any valid JavaScript expression that can be executed on the target page, for example, document.bgColor='red'. The expression is executed, using LiveConnect's JSObject.eval() method, in the context of the page or frame that is named in the target argument. If the target argument is null, then it is executed in the page or frame in which the Forms applet is embedded.

The second call, web.javascript_eval_function is a function and returns a varchar2 value. Both web.javascript_eval_expr and web.javascript_eval_function have the same functionality except that javascript_eval_expr does not send any return value from the Forms client to the Forms Services. If your application does not need a return value, use web.javascript_eval_expr. The additional network trip that is required to carry the return value from the Forms client to the Forms Services is eliminated.

To set the value of an HTML text item with the ID outside_field_id to the value of the Forms field called inside, you could write this PL/SQL code:

```
web.javascript_eval_expr('
document.getElementById("outside_field_id").value='
||:inside
);
```

Note that the PL/SQL string must use single quotes while JavaScript is flexible enough to use single or double quotes. Using double quotes inside the expression works without having to use escape sequences. You could also write a function in the Web page:

```
<SCRIPT>
    function set_field(field_id, myvalue){
        document.getElementById(field_id).value=myvalue;
    };
</SCRIPT>
```

To get the value of the outside field and assign it to the inside field, you could write the following PL/SQL code:

```
:inside:=web.javascript_eval_function('
     document.getElementById("outside_field_id").value
     ');
```

6.1.1 Why Call Events Outside of Oracle Forms?

In Oracle Forms 11*g*, the newly added JavaScript functionality allows you to integrate Forms with HTML-based application technologies in the Web browser. For example you can use JavaScript integration when the Forms-based application is required to integrate on the page with new functionality based on an HTML front end.

6.2 About JavaScript Events Calling into Oracle Forms

You can also allow JavaScript calls into Oracle Forms by using JavaScript in the Web page that hosts the Forms applet. There is new functionality available on the embedded Forms object in the DOM (Document Object Model) tree. You use JavaScript to do:

```
document.forms_applet.raiseEvent(event_name, payload);
```

The assumption here is that you have set the ID configuration variable to `forms_applet`.

When the surrounding Web page executes this JavaScript code, Oracle Forms fires a new type of trigger called `WHEN-CUSTOM-JAVASCRIPT-EVENT`. In this trigger there are only two valid system variables: `system.javascript_event_value` and `system.javascript_event_name`. These variables contain the payload and event name that were passed into Forms through the `raiseEvent` method. On calling the `raiseEvent` method, a trigger named `WHEN-CUSTOM-JAVASCRIPT-EVENT` is fired on the server side.

```
declare
    event_val varchar2(300):= :system.javascript_event_value;
begin
    if (:system.javascript_event_name='show') then
        handleShowEvent(event_val);
    elsif(:system.javascript_event_name='grab') then
  handleGrabEvent(event_val);
    else
        null;
    end if;
end;
```

This PL/SQL code recognizes two events: 'show' and 'grab'. Any other name is ignored.

6.2.1 Why Let Events Call into Oracle Forms?

You can synchronize an HTML based application, whether it is Java-based or otherwise, with a Forms-based application in the same hosting Web page. For example, you can use the HTML-based application to query data and use Forms to update it if, and only if, the user has the correct access privileges.

6.3 Integrating JavaScript and Oracle Forms

This section describes an example for integrating JavaScript in Oracle Forms application. To integrate JavaScript in Oracle Forms applications, perform the following steps:

1. Build a Forms application using the JavaScript events as described in Section 6.1, "About Oracle Forms Calling External Events" and Section 6.2, "About JavaScript Events Calling into Oracle Forms". Use the `:system.javascript_event_name` and `:system.javascript_event_value` in the `WHEN-CUSTOM-JAVASCRIPT-EVENT` trigger. Compile the module. For more information, refer to the Forms Builder Online Help.
2. Create an html file (for example, `test.html`) that the Forms servlet will use as a template when generating the HTML page used to start an Oracle Forms

application. Copy the file to the Forms configuration directory: `$ORACLE_INSTANCE/config/FormsComponent/forms/server`

3. Copy any required images, html files, JavaScript files, and css files to the following directory: $DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string2>/war/
4. Create an html file that uses the JavaScripts (for example, `js.html`) and invokes the servlet URL.
5. Using Enterprise Manager, create a new configuration section or modify an existing one and enable `enableJavascriptEvent`. Set baseHTMLjpi to `test.html`.
6. Using Enterprise Manager, edit the `default.env` file and add the directory where you saved the forms application to the environment variable `FORMS_PATH`.
7. Run the application by using the URL in your browser: `http://<localhost>:9001/forms/js.html`

6.4 Configuration of formsweb.cfg

The administrator of the Forms application can enable or disable JavaScript integration by setting the parameter `enableJavascriptEvent` in `formsweb.cfg` to "true" or "false". If `enableJavascriptEvent` is not set to true, then calls from JavaScript would be ignored. The applet_name parameter must be set to the value that is used by the HTML developer to reference the forms applet via document.<applet_name>.

The administrator can also set `JavaScriptBlocksHeartBeat` (default value is false) in `formsweb.cfg` to true. This blocks Form's `HEARTBEAT` during the time JavaScript is executed. If the JavaScript calls complete execution before the FORMS_TIMEOUT period, setting JavaScriptBlocksHeartBeat to true provides an increase in performance by avoiding additional network messages.

Note that if JavaScriptBlocksHeartBeat is set to true, Forms would abnormally terminate if the time taken for executing a JavaScript is more than FORMS_TIMEOUT.

6.5 Configuration of Environment Variables

An environment variable called `FORMS_ALLOW_JAVASCRIPT_EVENTS` in `default.env` is also used to enable or disable JavaScript integration. By default, the value of the variable is true. If this is set to false, then JavaScript integration is not enabled for any Forms aplication that uses that instance of `default.env`, no matter what value is set for `enableJavascriptEvent` in `formsweb.cfg`.

7

Enhanced Java Support

This chapter contains the following sections:

7.1 Overview

Oracle Forms provides Java classes that define the appearance and behavior of standard user interface components such as buttons, text areas, radio groups, list items, and so on. A Forms pluggable Java component (PJC) can be thought of as an extension of the default Forms client component. When you create a PJC, you write your own Java code to extend the functionality of any of the provided default classes.

7.1.1 Dispatching Events from Forms Developer

In addition to extending the standard Forms user interface components, you can also create a PJC that includes Java Swing user interface components in your form. A pluggable Java component extends a class provided by Forms, that is, `oracle.forms.ui.VBean`, and lives in the Bean Area as seen on the Forms canvas. The Bean Area does not have its own user interface, but rather is a container. On the layout editor or on a canvas, you see only an empty rectangle until you associate an implementation class with it and add some user interface components.

In earlier releases of Oracle Forms, Forms user interface components implemented the IView interface. However, it did not have any special method to add or remove CustomListener from the pluggable Java component or the view. In Oracle Forms 11*g*, you can add or remove CustomListener in the IView interface.

7.1.2 Dispatching Events to Forms Services

Oracle Forms 11*g* makes it easier to dispatch CustomEvent along with parameters and payloads. Since JavaBean classes do this by exposing the public method `dispatchCustomEvent`, you need to add the same method for your PJC. You call the `dispatchCustomEvent` method from the PJC to dispatch the CustomEvent.

Since CustomEvent is usually associated with parameters, Forms provides a way to add them. In a JavaBean, you can use the `getHandler().setProperty()` method to set the parameters. Users must be able to do the same for PJC. For more information, see Section 7.2.2, "About the Custom Item Event Trigger at Runtime".

7.2 About Custom Item Event Triggers

In Oracle Forms 11*g*, you can add the WHEN-CUSTOM-ITEM-EVENT trigger to items at design time and code the pluggable Java components so that the trigger can be fired at runtime. This trigger fires whenever a JavaBean custom component in the form causes the occurrence of an event. You can use a WHEN-CUSTOM-ITEM-EVENT trigger to respond to a selection or change of value of a custom component. The system variable SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS stores a parameter name that contains the supplementary arguments for an event that is fired by a custom control. Control event names are case sensitive.

7.2.1 Adding the When-Custom-Item-Event Trigger at Design Time

The most common way of adding a trigger to an item is by clicking the Create button in the Object Navigator toolbar in Oracle Forms Developer, while the focus is on the Trigger node, or by pressing the corresponding shortcut key. Forms Developer presents to you a list of available triggers at that level or for that item.

Another way of adding some of the commonly used triggers is by right-clicking the trigger node of the item in the Object Navigator. Then, select one of the triggers listed in the smart Triggers menu.

For more information on working with triggers, see the Oracle Forms Developer online help.

7.2.2 About the Custom Item Event Trigger at Runtime

In Oracle Forms 11*g*, pluggable Java components can raise the WHEN-CUSTOM-ITEM-EVENT trigger. This enhanced trigger provides greater control over the content of the communication between the client and server.

The Forms client dispatches CustomEvent through the pluggable Java component, which fires the WHEN-CUSTOM-ITEM-EVENT trigger on the Forms Services. The WHEN-CUSTOM-ITEM-EVENT trigger provides a simple way to retrieve the event name and parameter values that are passed from the client pluggable Java component through CustomEvent. The event name is stored in SYSTEM.CUSTOM_ITEM_EVENT; parameters (name and value) are stored in SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS.

The Forms Built-in get_parameter_attr is used to retrieve the values and different parameters from SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS. The supported datatype for the values or payloads that are returned from get_parameter_attr is a VARCHAR2 string.

7.2.3 Example: A Java class for a Push Button

In this example, a Java class is created for a push button that enables selecting a client file using the File Open option and returns the path to the server.

1. Create a Java class for a push button with simple PJC code such as:

```
 // MyButtonPJC.java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFileChooser;
import oracle.forms.ui.CustomEvent;
import oracle.forms.ui.VButton;
import oracle.forms.properties.ID;
public class MyButtonPJC extends VButton implements ActionListener
```

```
 {
   private static final ID CLIENT_SELECTED_FILE = ID.registerProperty("CLIENT_
SELECTED_FILE");
   public MyButtonPJC()
   {
     addActionListener(this);
   }
   public void actionPerformed(ActionEvent event)
   {
     JFileChooser fc = new JFileChooser();
     if(fc.showOpenDialog(getHandler().getApplet()) == JFileChooser.APPROVE_
OPTION)
     {
       CustomEvent ce = new CustomEvent(getHandler(), "MyButtonPJC_Event");
       ce.setProperty(CLIENT_SELECTED_FILE,
fc.getSelectedFile().getAbsolutePath());
       this.dispatchCustomEvent(ce);
     }
   }
   public void destroy()
   {
     removeActionListener(this);
     super.destroy();
    }
}
```

2. Ensure CLASSPATH variable is defined in the environment and `$ORACLE_HOME/forms/java/frmall.jar` is added to it.
3. Compile the Java class. For ease of creating the jar later, place the output class files in a separate directory by using the `-d <output-directory>` option of the javac (java compiler).
4. Navigate to the output directory and create a jar file, for example, MyButtonPJC.jar, containing the generated class files by using the command

   ```
   jar cvf <jar-file-path> *
   ```

5. MyButtonPJC.jar needs to be signed before deploying in Forms applet. You can use `sign_webutil.sh` (`sign_webutil.bat` in Windows) that is available in the directory `$ORACLE_INSTANCE\bin` to sign the jar file. For more information, see Forms Builder Online Help.
6. Copy MyButtonPJC.jar to `$ORACLE_HOME/forms/java` directory.
7. Add the path of MyButtonPJC.jar to the FORMS_BUILDER_CLASSPATH. This makes the class files in that jar available in Forms Builder.
8. Add the push button on the layout in the Forms application.
9. In Property Palette of the push button, set MyButtonPJC as the implementation class.
10. Add `WHEN-CUSTOM-ITEM-EVENT` trigger to the push button.
11. Add the following PL/SQL code to the `WHEN-CUSTOM-ITEM-EVENT` trigger of the push button. This code handles the CustomEvent dispatched by the PJC and then extracts the parameters in the event.

    ```
    declare
       filePath VARCHAR2(1024);
       dataType      PLS_INTEGER;
     begin
       Message('Custom Event Name='||:SYSTEM.CUSTOM_ITEM_EVENT);
    ```

```
    get_parameter_attr(:SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS,'CLIENT_SELECTED_
FILE',dataType, filePath);
    Message('The selected client file path is '|| filePath);
  end;
```

12. Add MyButtonPJC.jar to the list of comma-separated jars (only jar file name, not the full path) in the `archive` parameter in Forms configuration file (`formsweb.cfg`). This ensures that the jar file is loaded in Forms applet on the client side.

8

Working with Server Events

This chapter contains the following:

8.1 About Oracle Forms and Server Events

With the exception of timers, most events in Oracle Forms occur from some kind of user interaction. In previous versions of Oracle Forms, there was no easy support to receive an external event if it could not be bound to the Form's graphical user interface. Forms clients had to use techniques such as polling through a great deal of coding to respond to these events to deal with external events that it did not initiate.

With Oracle Forms 11*g* and Oracle Database, you can handle external events, such as asynchronous events, by using the database queue. Note that in order to work with database queues in Oracle Forms 11*g* you must be using Oracle Database 10*g* Release 2 or later. Oracle Streams Advanced Queuing (AQ), an asynchronous queuing feature, enables messages to be exchanged between different programs. AQ functionality is implemented by using interfaces such as DBMS_AQ, DBMS_AQADM, and DBMS_AQELM, which are PL/SQL packages. For more information about Advanced Queuing, see the *Oracle Streams Advanced Queuing User's Guide* at `http://www.oracle.com/technology/documentation/index.html`.

In general, the steps required to integrate events and database queues are:

Database

- Create a queue table: Define the administration and access privileges (AQ_ADMINISTRATOR_ROLE, AQ_USER_ROLE) for a user to set up advanced queuing. Define the object type for the payload and the payload of a message that uses the object type. Using the payload, define the queue table.
- Create a queue: Define the queue for the queue table. A queue table can hold multiple queues with the same payload type.
- Start the queue: Enable enqueue/dequeue on the queue.

- Enqueue a message: Write messages to the queue using the DBMS_AQ.ENQUEUE procedure.

Forms Builder

- Create an event object: Create a new event in the Events node in the Object Navigator in the Forms Builder.
- Subscribe the event object to the queue: The name of the queue is specified in the Subscription Name property.
- Code necessary notification: Write the event handling function, which is queued up for execution by Forms and is executed when the server receives a request from the client. Write the trigger code for the When-Event-Raised trigger that is attached to the Event node.

Forms Services

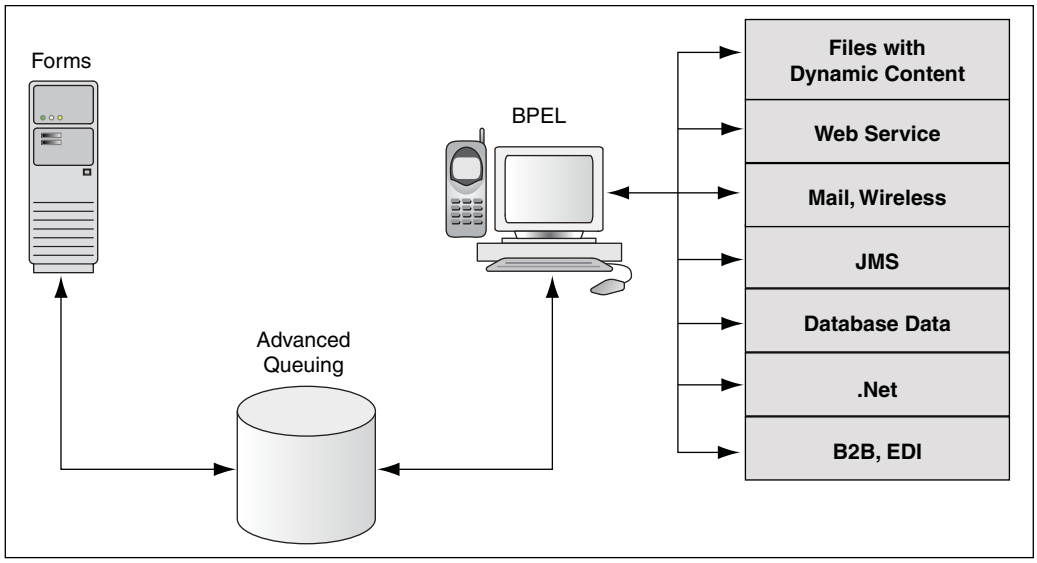
- Run the form and register the subscription
- Invoke the When-Event-Raised trigger upon event notification

In earlier versions of Forms, handling external events was only possible through custom programming, usually done in Java with the help of Forms' Java Bean support. In Oracle Forms 11*g* it is possible to call into Forms from any technology that can interface with Advanced Queuing (AQ), for example Java Messaging (JMS).

Figure 8–1 shows the flow of events that take advantage of the improved integration of the different components your application might work with. In the left side of the image, the Oracle Forms has two-way communication with the AQ functionality of Oracle Database. In the center of the image, the AQ function of Oracle Database also has two-way communication with the possible outside events that can trigger internal Forms events. In the right side of the image, these external events can include technologies such as files with dynamic content, Web services, mail, JMS, or database content that interact with BPEL processes which in turn interact with AQ. BPEL, however, is not necessary. JMS, as an example, can interact with AQ directly without having to go through BPEL.

Note: Third party tools such as antivirus and security software may prevent Advanced Queuing from working correctly with Oracle Forms. As a workaround, turn off any third party security tools.

Figure 8–1 Oracle Forms Handles Outside Events with Advanced Queueing in Oracle Database



8.2 Creating Events

Oracle Forms Developer provides a declarative environment for creating and managing event objects. For known external events, Forms Developer provides a list of available events that can be subscribed to. The property of the event object can be set at runtime or at design time. The ability to end a subscription to a particular external event is also provided through a dynamic setting of the event object property.

Most of the new event functionality is also available through standard Oracle interfaces. Both client and server-side PL/SQL provide all the necessary functionality to create, subscribe, and publish a database event. Oracle Forms provides a declarative and user-friendly way of registering a database event. Oracle Forms provides a standard way of responding to the event by hiding most of the complexity from end-users.

8.3 Subscribing to Events

The Forms Services gets notified when events it has registered interest in are added to the event queue. Registration is done either when the runtime starts up or when connecting to the database, depending on the type of the event. For database events, the type of the event queue (persistent or non-persistent) is also saved as part of the event creation.

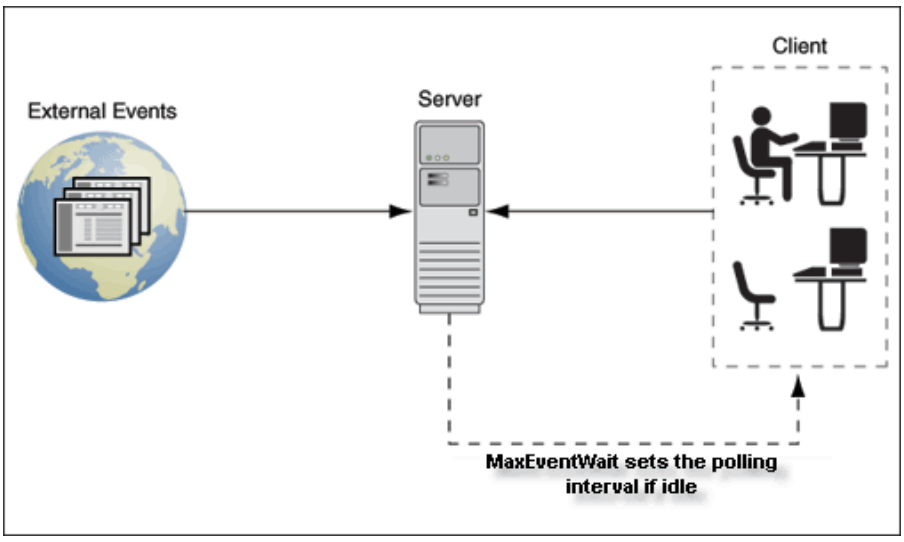
8.4 Event Propagation

Figure 8–2 shows a situation where a Forms client is idle. Since Oracle Forms is driven by the HTTP protocol, which is a request/response protocol only, nothing can change on the client if the client is idle. A new applet property `MaxEventWait`, expressed in milliseconds, governs how long the application should wait before checking for an event. In other words, you can specify how often the client should send a request to the server, thus causing the execution of the PL/SQL that is specified as a response to an event.

Note, however, that, on the server-side, Forms Services receives all the events without polling. However, the server does not start running the `WHEN_EVENT_RAISED` triggers

until it receives the notification from the Forms Client (because of the HTTP request/reply paradigm of the Forms Client and hence the need for the `MaxEventWait` property).

Figure 8–2 Notification flow with idle or active clients



8.4.1 About the When-Event-Raised Trigger

Oracle Forms responds to or fires a trigger in response to a variety of events. For both Forms Developer and internal events, Forms provides entry points in terms of triggers so that an application developer can associate and execute some code in response to an event.

For example, a defined trigger is attached to a specific object in a form. The object to which a trigger is attached defines the *scope* of the trigger. For example, the `WHEN-BUTTON-PRESSED` trigger corresponds to the Button Pressed event which occurs when an operator selects a button. The name of the trigger establishes the association between the event and the trigger code. When a user clicks on a button, Forms responds by executing the code in the `WHEN-BUTTON-PRESSED` trigger.

This new event object has a corresponding trigger defined at the event object level. The `WHEN-EVENT-RAISED` trigger fires in response to the occurrence of a database event for which it has a subscription. The firing of the new trigger is similar to the internal processing of triggers. However, the source of the event is, in this case, an external event such as a database event (firing as a result of an operation) and not the result of any user interaction with forms or as a result of an internal form processing.

8.4.2 About Trigger Definition Level and Scope

Oracle Forms triggers are usually attached to a specific object, such as an item, block, or Form. The object to which a trigger is attached determines the trigger's *definition level* in the object hierarchy. A trigger's definition level determines the trigger's *scope*. The scope of a trigger is its domain within the Forms object hierarchy, and determines where an event must occur for the trigger to respond to it. Although the `WHEN-EVENT-RAISED` trigger is attached to an event object, it has an application level scope because of the nature of the server-centric events. When the event notification is invoked as a result of an asynchronous callback mechanism for registered database events, any number of forms running within that application and with a subscription

for that event receive the notification. This alleviates the need for the application developer to code complex logic to deal with the event.

There is also a Form-level scope so that the event will only be handled if the application is running the specific form from where the event is defined.

8.5 Publishing Database Events

You use the standard PL/SQL interface for publishing a database event from Forms. For example, you can publish the `SalaryExceed` event by calling the enqueue interface and providing all the necessary arguments. You can also call a stored procedure to perform this task.

The following program unit can be called from a WHEN-BUTTON-PRESSED trigger by passing the queue name. Depending on how you have defined the queue in the database, a commit might or might not be necessary to actually publish the event. The following sample code will not actually publish the event since there is no commit issued.

```
Declare
  msgprop      dbms_aq.message_properties_t;
  enqopt       dbms_aq.enqueue_options_t;
  enq_msgid    raw(16);
  payload      raw(10);
  correlation  varchar2(60);
begin
    payload := hextoraw('123');
    correlation := 'Jones';
    enqopt.visibility := dbms_aq.IMMEDIATE;
    msgprop.correlation := correlation;
    DBMS_AQ.ENQUEUE( queue, enqopt, msgprop, payload, enq_msgid);
end;
```

For more information about database events, see *Oracle Database PL / SQL Reference*.

8.6 About Application Integration Between Forms

Many enterprise applications are made of a large number of forms which are defined to perform specific tasks such as purchasing, accounting, and sales force management. These applications may also interact with other non-Forms based applications as part of performing a task. The need to provide an integration model where an enterprise can easily integrate its applications (including passing data) with those of its partners, suppliers, and distributors is extremely important.

In previous releases, Oracle Forms attempted to integrate loosely coupled applications through mechanisms ranging from using user_exit calls and some polling via timers to using pluggable Java components. These methods are all useful in some limited circumstances, but they do not provide a formal infrastructure for enterprise application integration.

Apart from the deployment concerns and performance issues, the main reason why these methods do not fully integrate applications is that the integration is only provided through Forms Developer as almost all events are bound to Forms visual components. Also, the communication with the Forms Services is always initiated by the Forms client via a request-reply model.

To provide better support for application integration, Oracle Forms 11*g* supports synchronous and asynchronous server-centric events.

8.6.1 About Synchronous Communication

Synchronous communication follows a request-reply paradigm, where a program sends a request to another program and waits until the reply arrives. HTTP follows this paradigm. This model of communication (also called online or connected) is suitable for programs that need to get the reply before they can proceed with their work. Traditional client-server architectures are based on this model. Earlier releases of Oracle Forms client-server architecture is also an example of this model. One of the drawbacks of the synchronous model of communication is that all the programs must be available and running for the application to work. In the event of network or machine failure, programs cease to function. For example, if the Forms Services dies, the Forms client ceases to function as well. The synchronous communication model is also in use when the Forms Services interacts with other systems such as PL/SQL or the database. The Forms system would be blocked waiting for the current operation to end before continuing with its work. Another drawback of synchronous communication is that the calling program has to wait for a response and unexpected events cannot be handled without first polling for them.

8.6.2 About Asynchronous Communication

Asynchronous communication is when a user or form places a request in a queue and then proceeds with its work without waiting for a reply or when an asynchronous event is received without any initial request. Programs in the role of consumers retrieve requests from the queue and act on them. This model is well-suited for applications that can continue with their work after placing a request in the queue because they are not blocked waiting for a reply. It is also suited to applications that can continue with their work until there is a message to retrieve.

Oracle Forms 11*g* supports asynchronous communication with the help of database events. A thin queuing mechanism provides the mechanism for asynchronous events. The queue is checked for messages once there are no more current operations to be performed.

For example, an application might require data to be entered or an operation executed at a later time, after specific conditions are met. The recipient program retrieves the request from the queue and acts on it.

8.6.3 Configuring Asynchronous Communication

Oracle Forms uses a polling technique at the application level. The client polls the server for an update after specified intervals of time. The frequency of polling can be modified using the parameters - `MaxEventWait` and `HEARTBEAT`. A higher frequency of polling may ensure that a client polls the server more frequently for updates; however, this may result in consumption of considerable resources.

The frequency value for polling is set in `formsweb.cfg`. The value assigned to this constant is in milliseconds and is a positive number.

In the absence of the configuration file setting, the current Oracle Forms `HEARTBEAT` setting is used. However, special attention and care should be made with regards setting and using of `MaxEventWait`. In a default setting where `MaxEventWait` is not set, the `HEARTBEAT` mechanism is used for polling. The default delay when the HEARTBEAT mechanism is used is two minutes. You can set the `MaxEventWait` (which is in milliseconds) to a value smaller than the `HEARTBEAT` for faster response.

For more information on configuring these parameters using the Enterprise Manager, see Chapter 4.2.4, "Managing Parameters".

9

Using Forms Services with Oracle Single Sign-On

This chapter contains the following sections:

9.1 Overview

Oracle Forms Services applications can run in a Single Sign-on environment using Oracle Application Server Single Sign-On Server (OracleAS Single Sign-On) and Oracle Internet Directory to store user name and password information. OracleAS Single Sign-On is designed to work in Web environments where multiple Web-based applications are accessible from a browser. Without OracleAS Single Sign-On, each user must maintain a separate identity and password for each application they access. Maintaining multiple accounts and passwords for each user is unsecured and expensive.

OracleAS Single Sign-On Server 10*g* enables an application to authenticate users by means of a shared authentication token or authentication authority. That means that a user authenticated for one application is automatically authenticated for all other applications within the same authentication domain.

Forms applications use OracleAS Single Sign-On Server 10*g* only for obtaining database connection authentication. Once this connection is made, interaction with OracleAS Single Sign-On Server 10*g* no longer occurs. Exiting a Forms application does not perform an OracleAS Single Sign-On Server 10*g* logout. Conversely, logging out of an OracleAS Single Sign-On Server session does not terminate an active Forms session. The database session exists until the Forms Runtime (for example, `frmweb.exe`) on the server terminates, usually by explicitly exiting the form.

OracleAS Single Sign-On Server 10*g* can be used to authenticate other applications that are not Oracle products, for example, custom-built Java EE applications.

Oracle Forms applications integrate into a company's OracleAS Single Sign-On Server architecture based on OracleAS Single Sign-On Server and the Oracle Internet Directory. Oracle Forms Services provides out-of-the box support for single sign-on for as many Forms applications as run by the server instance with no additional coding required in the Forms application.

Note: Refer to the Section 3.4.2, "Forms Single Sign-On on Mozilla 3.x" for more information on browser support for Forms and single sign-on.

Note: Oracle Forms Services applications runs in a Single Sign-on environment using the following OID and SSO combinations:

- Oracle Internet Directory 10*g* (10.1.2.3) with Oracle Single Sign-On 10*g* (10.1.2.3)
- Oracle Internet Directory 10*g* (10.1.4.3) with Oracle Single Sign-On 10*g* (10.1.4.3)
- Oracle Internet Directory 11*g* (11.1.1) with Oracle Single Sign-On 10*g* (10.1.4.3)

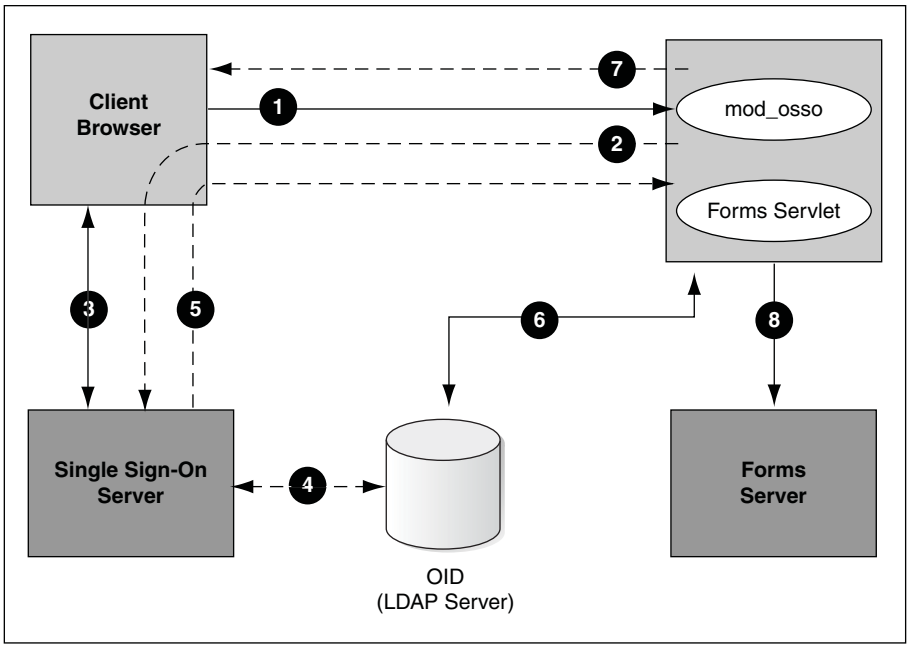
For more information on OracleAS Single Sign-On, see the *Oracle Application Server Single Sign-On Administrator's Guide* on OTN.

For more information on Oracle Internet Directory, see *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*.

9.1.1 Authentication Flow

Figure 9–1 describes the authentication flow of OracleAS Single Sign-On Server 10*g* support in Oracle Forms the first time the user requests an application URL that is protected by OracleAS Single Sign-On Server 10*g* :

Figure 9–1 Authentication Flow for First Time Client Request



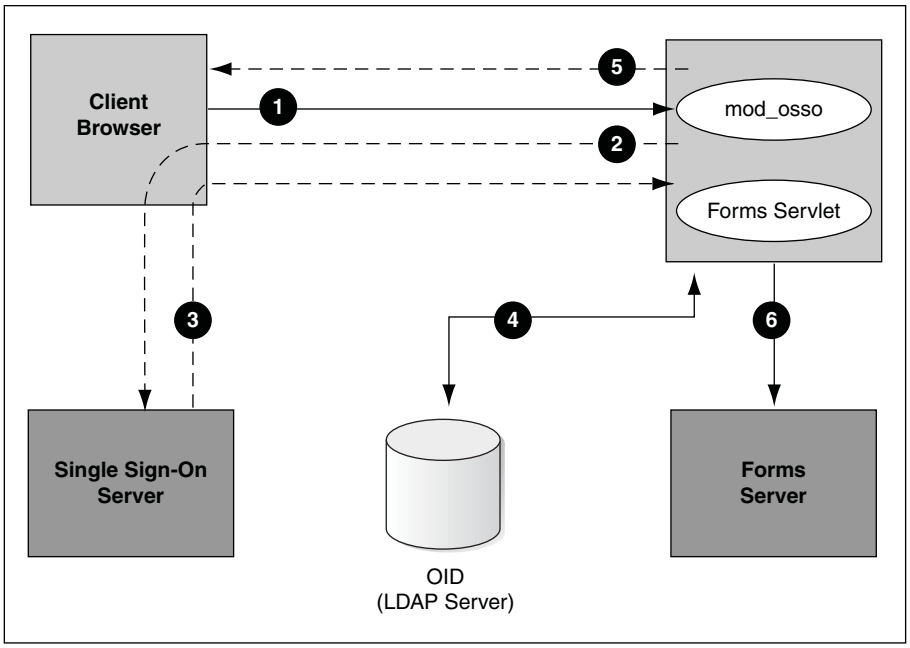
1. In the upper left of the image, the user requests a Forms URL similar to `http(s)://<hostname>:<port>/forms/frmservlet?config=<application>&...`

   Note: Use the HTTP or Web Cache port number in the Forms URL for Forms applications that use single sign-on. The Forms URL is similar to `http://<host name>:<http port>/forms/frmservlet?config=ssoapp` where ssoapp is the name of the section in forms configuration file with single sign-on (`ssoMode`) enabled.

2. The Forms servlet redirects the user to the OracleAS Single Sign-On Server login page, indicated on the bottom left of the image.
3. The user provides user name and password through the login form.
4. The password is verified through Oracle Internet Directory (LDAP Server), shown at the center of the image.
5. The user is redirected to the URL with `sso_userid` information, indicated in the upper right of the image.
6. The Forms servlet retrieves the database credentials from Oracle Internet Directory.
7. The Forms servlet sets the user ID parameter in the Runform session and permits the applet to connect to the Forms listener servlet.
8. The Forms servlet starts the Forms server.

Figure 9–2 describes the authentication flow of single sign-on support in Oracle Forms Services when a user, authenticated through another partner application, requests an application that is protected by OracleAS Single Sign-On Server.

Figure 9–2 Authentication Flow for Subsequent Client Requests



1. The user requests the Forms URL, as shown in the upper left side of the image.
2. The Forms servlet redirects the user to the OracleAS Single Sign-On Server server and its login page, indicated on the bottom left of the image.
3. The user is redirected to the URL with the `sso_userid` information.
4. The Forms servlet retrieves the database credentials from Oracle Internet Directory, as shown in the center of the image.
5. The Forms servlet sets the user ID parameter in the Runform session and the applet connects to the Forms listener servlet.
6. The Forms servlet starts the Forms server, shown on the bottom right of the image.

9.2 Available Features with OracleAS Single Sign-On, Oracle Internet Directory and Forms

The following features and enhancements are available with this release of Oracle Forms Services:

- Section 9.2.1, "Dynamic Resource Creation When A Resource Is Not Found In Oracle Internet Directory"
- Section 9.2.2, "Support for Dynamic Directives With Forms and OracleAS Single Sign-On"
- Section 9.2.3, "Support for Database Password Expiration for Forms Running with OracleAS Single Sign-On"

9.2.1 Dynamic Resource Creation When A Resource Is Not Found In Oracle Internet Directory

In single-sign on mode, when a user tries to connect to a database using Forms, the user is authenticated by mod_osso in combination with the OracleAS Single Sign-On Server and Oracle Internet Directory. Once the user is authenticated, the user is

directed to the Forms servlet which takes the user's request information containing the single sign-on user name. The user name and the application name build a unique pair that identifies the user's resource information for this application in Oracle Internet Directory.

When an authenticated Forms user has neither the resource for a particular application that is being requested nor a default resource in Oracle Internet Directory, then the user is redirected to the self-service console page of Oracle Internet Directory/DAS to dynamically create them. After creating the resource, the user is redirected back to the original Forms request URL.

The way Forms Services handles the missing resource information can be customized by the application or Forms Services administrator. The following options are available:

- Allow dynamic resource creation (default)
- Redirect the user to a pre-defined URL as specified by the ssoErrorUrl parameter
- Display the Forms error message

The redirection URL is provided by the system administrator in the Forms configuration files and should be either absolute or relative.

9.2.2 Support for Dynamic Directives With Forms and OracleAS Single Sign-On

Enforcing single sign-on in Forms is done within the `formsweb.cfg` file. The single sign-on parameter, `ssoMode`, when set to TRUE, indicates that the application requires authentication by OracleAS Single Sign-On Server.

This parameter allows a Forms Services instance to handle both application types, ones protected by database password and ones protected by OracleAS Single Sign-On Server. Because single sign-on is configured in the `formsweb.cfg` file, Enterprise Manager Fusion Middleware Control can be used to manage this aspect of authentication.

9.2.3 Support for Database Password Expiration for Forms Running with OracleAS Single Sign-On

In previous releases of Oracle Forms, changing a database password would be successful, but the changes (including expirations) would not propagate to Oracle Internet Directory.

In Oracle Forms Services 11*g*, if the database password has expired, the *Forms Services* application, running in single sign-on mode, is used to renew it, the new password entered by the user is used to update the Resource Access Descriptor (RAD) in Oracle Internet Directory for this application. This feature ensures that authenticating a Forms user via OracleAS Single Sign-On Server with Forms continues to work even when the user's database password has changed. However, if password changes are made in SQL*PLUS, and not in Oracle Forms, the database connect string is not updated in Oracle Internet Directory.

9.3 OracleAS Single Sign-On Components Used By Oracle Forms

The following software components in Oracle Fusion Middleware are involved when running Forms applications in single sign-on mode:

- OracleAS Single Sign-On Server - an authentication service in Oracle Fusion Middleware that uses Oracle Internet Directory to store user names and passwords
- `mod_osso` - The HTTP module `mod_osso` simplifies the authentication process by serving as the sole partner application to the OracleAS Single Sign-On Server, rendering authentication transparent for applications. Oracle Forms Services and Oracle Reports Services use `mod_osso` to register as partner applications with the OracleAS Single Sign-On Server.
- Oracle Internet Directory - A LDAP v3 compliant directory server that stores user login information. An LDAP server is a special database that is optimized for read access.
- Forms servlet - The Oracle Forms Services component that accepts the initial user request to start a Forms application. The Forms servlet detects if an application requires OracleAS Single Sign-On Server, directs the request to the OracleAS Single Sign-On Server and accesses the Oracle Internet Directory to obtain the database connect information.
- `formsweb.cfg` - The Forms configuration file that contains the parameters to enable a Forms application for single sign-on. The `formsweb.cfg` file is located in the $DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config directory.

9.4 Enabling OracleAS Single Sign-On for an Application

Oracle Forms applications are configured using a central configuration file, the `formsweb.cfg` file in the $DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config directory. The recommended method of managing `formsweb.cfg` file is using Fusion Middleware Control.

Single sign-on and error handling are defined by the following parameters in the `formsweb.cfg` file:

- `ssoMode` [true|false]
- `ssoProxyConnect` [yes|no]
- `ssoDynamicResourceCreate` [true|false]
- `ssoErrorUrl` [String URL]
- `ssoCancelUrl` [String URL]

These Oracle Forms parameters in the `formsweb.cfg` file are set in the **User Parameter** section, which define the behavior for all Forms applications run by the server. These parameters can also be set in a **Named Configuration**, which define the settings for a particular application only. A single sign-on parameter set in a Named Configuration section overrides the same parameter set in the **User Parameter** section.

To enable single sign-on for an application:

1. Start Fusion Middleware Control.
2. Select **Web Configuration** from the **Forms** menu.
3. Select the row that lists the configuration section for your application.
4. In the Section region, select **sso** in the **Show** drop down list.
5. In the Section region, select the row containing `ssoMode`.

6. In the **Value** field, enter `true`.
7. Click **Apply** to update the `formsweb.cfg` file.

   Single sign-on is now enabled for the selected application.

To disable single sign-on for an application:

1. Select **Web Configuration** from the **Forms** menu.
2. Select the row that lists the configuration section for your application.
3. In the Section region, select **sso** in the **Show** drop down list.
4. In the Section region, select the row containing `ssoMode`.
5. In the **Value** column, enter `false`.
6. Click **Apply**.

   Single sign-on is now disabled for the selected application.

9.4.1 ssoMode

The `ssoMode` parameter enables a Forms Services application to connect to OracleAS Single Sign-On Server. By default, Oracle Forms applications are not configured to run in single sign-on mode. The `ssoMode` parameter can be set in two places in the `formsweb.cfg` file:

- By setting `ssoMode` in the default section of `formsweb.cfg` with a value of `true` which allows all applications to run in single sign-on mode by this Forms Services instance
- By setting the `ssoMode` parameter in a named configuration of an Oracle Forms application which enables or disables single sign-on only for this particular application, for example:

```
[myApp]
form=myFmx
ssoMode=true
```

9.4.2 ssoProxyConnect

The `ssoProxyConnect` parameter enables a user to control when Oracle Forms should use a proxy connection to the database and when it should not. The `ssoProxyConnect` parameter can be set in two ways:

- By setting `ssoProxyConnect` in the default section of `formsweb.cfg` with a value of `yes` which allows all applications to run in single sign-on mode by this Forms Services instance
- By passing the `ssoProxyConnect` parameter in the URL at runtime, for example `http://<host>:<port>/?config=myapp&.....&ssoProxyConnect=yes`

9.4.3 ssoDynamicResourceCreate

The `ssoDynamicResourceCreate` parameter is set to `true` by default which allows the user to create a Resource Access Descriptor (RAD) entry in Oracle Internet Directory to run the application if this resource entry does not exist. The Web page used is a standard form provided by the Oracle Delegated Administration Services. This Web page cannot be customized as it is not owned by Oracle Forms.

Allowing dynamic resource creation simplifies Oracle Internet Directory administration because there is no longer the need for an administrator to create user RAD information in advance. The `ssoDynamicResourceCreate` parameter can be set as a system parameter in the `formsweb.cfg` file or as a parameter of a named configuration. Because the default is set to `true`, this parameter may be used in a named configuration for a specific application to handle a missing RAD entry differently from the default.

Note that enabling an application for single sign-on with the value of the `ssoDynamicResourceCreate` parameter set to `false`, while not specifying a value for the `ssoErrorURL`, causes Oracle Forms to show an error message if no RAD resource exists for the authenticated user and this application.

Since not all administrators want their users to create resources for themselves (and potentially raising issues with Oracle Internet Directory), these parameters allow administrators to control Oracle Internet Directory resource creation. Although the default behavior is to direct users to an HTML form that allows them to create the resource, the administrator can change the setting and redirect the user to a custom URL.

For the configuration section for the Forms application, you need to set these parameters:

```
[myApp]
form=myFmx
ssoMode=true
ssoDynamicResourceCreate=false
```

For information about setting these parameters through Enterprise Manager Fusion Middleware Control, see Section 4.2.4, "Managing Parameters".

9.4.4 ssoErrorURL

The `ssoErrorURL` parameter allows an administrator to specify a redirection URL that handles the case where a user RAD entry is missing for a particular application. This parameter only has effect if the `ssoDynamicResourceCreate` parameter is set to `false`, which disables the dynamic resource creation behavior. The `ssoErrorURL` parameter can be defined in the default section and as a parameter in a named configuration section. The URL can be of any kind of application, a static HTML file, or a custom Servlet (JSP) application handling the RAD creation, as in the example below.

```
[myApp]
form=myFmx
ssoMode=true
ssoDynamicResourceCreate=false
ssoErrorURL=http://example.com:7779/servlet/handleCustomRADcreation.jsp
…
```

9.4.5 ssoCancelUrl

The `ssoCancelURL` parameter is used in combination with the dynamic RAD creation feature (`ssoDynamicResourceCreate= true`) and defines the URL that a user is redirected to if the user presses the cancel button in the HTML form that is used to dynamically create the RAD entry for the requested application.

9.4.6 Accessing Single Sign-on Information From Forms

Optionally, if you need to work with OracleAS Single Sign-On Server authentication information in a Forms application, the GET_APPLICATION_PROPERTY() Built-in can be used to retrieve the following single sign-on login information: single sign-on user ID, the user distinguished name (dn), and the subscriber distinguished name (subscriber dn)

```
authenticated_username := get_application_property(SSO_USERID);
userDistinguishedName := get_application_property(SSO_USRDN);
subscriberName := get_application_property(SSO_SUBDN);
config := get_application_property(CONFIG).
```

Note: `config` can be obtained even in non-SSO mode.

9.4.7 Registering Oracle HTTP Server with OracleAS Single Sign-On Server

Perform these steps if you chose to install and configure Forms in non-SSO mode and later need to enable SSO. Perform the following steps to register the module `mod_osso` in the WebTier OHS with the OracleAS Single Sign-On Server as a partner application.

1. Generate and copy the `osso.conf` file as mentioned in steps 3, and 4 of "To re-associate an OID Host with a Forms Application".
2. Create a `mod_osso.conf` file under `$ORACLE_INSTANCE/config/OHS/<OHS_INSTANCE>/moduleconf` directory. The contents of the file should look similar to this:

```
 LoadModule osso_module ${ORACLE_HOME}/ohs/modules/mod_osso.so
<IfModule mod_osso.c>
OssoIpCheck off
OssoSecureCookies off
OssoIdleTimeout off
OssoConfigFile osso.conf
#
# Insert Protected Resources: (see Notes below for
# how to protect resources)
#
#______-
#
# Notes
#
#______-
#
# 1. Here's what you need to add to protect a resource,
#    e.g. <ApacheServerRoot>/htdocs/private:
#
<Location /private>
require valid-user
AuthType Osso
</Location>

</IfModule>

#
# If you would like to have short hostnames redirected to
# fully qualified hostnames to allow clients that need
# authentication via mod_osso to be able to enter short
```

```
# hostnames into their browsers uncomment out the following
# lines
#
#PerlModule Apache::ShortHostnameRedirect
#PerlHeaderParserHandler Apache::ShortHostnameRedirect
```

3. Add the following lines to the beginning of `forms.conf` file.

```
<IfModule !mod_osso.c>
LoadModule osso_module  ${ORACLE_HOME}/ohs/modules/mod_osso.so
</IfModule>
<IfModule mod_osso.c>
OssoHTTPOnly off
</IfModule>
```

4. Associate the OID Host in Enterprise Manager as given in the topic *"To Associate OID Host with a Forms Application"* of the Section 9.7, "Configuring Oracle Internet Directory".
5. Restart the Oracle WebLogic Managed Server (WLS_FORMS) and the front-end OHS for the changes to take effect.

9.5 Integrating Oracle Forms and Reports

Oracle Reports is installed with OracleAS Single Sign-On Server enabled.

The best practice for Oracle Forms applications calling integrated Oracle Reports is to use the Oracle Forms Built-in, `RUN_REPORT_OBJECT`.

When requesting a report from a SSO-enabled Oracle Forms application, the authenticated user's SSO identity is implicitly passed to the Reports Server with each call to RUN_REPORT_OBJECT built-in. The SSO identity is used to authenticate the user to the Reports Server for further authorization checking, if required.

A Forms application running in non-SSO mode can run a report on a SSO-secured Reports Server, but fails if the Reports Server requires authorization. Also, users must provide their SSO credentials when retrieving the Reports output on the Web.

For more information on enabling single sign-on in Forms, see Section 9.4, "Enabling OracleAS Single Sign-On for an Application".

For more information on configuring single sign-on in Reports, refer to the *Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services*.

For more information about integrating Oracle Forms and Oracle Reports, see the white paper *Integrating Oracle Forms 11g and Oracle Reports 11g* at `http://www.oracle.com/technology/products/forms/`.

9.5.1 Forms and Reports Integration in non-SSO mode

Prior to 11g Release 1 (11.1.1), Oracle Reports generated sequential job IDs, making it easy to predict the job ID. This meant that unauthorized or malicious users could potentially view the job output using GETJOBID through rwservlet to obtain job output that belongs to another user. In 11*g*, Oracle Reports generates random and non-sequential job IDs to make it impossible to predict the job ID for a particular job. Only the user who runs a report from Oracle Forms Services is able to see its output. Other users should not be able to see the report output as job IDs are random non-sequential numbers.

For a non-secure Reports Server, the user ID and password for administrators can be set in the identifier element of the Reports Server configuration file.

For more information on configuring the access levels for the users, refer to the *Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services*.

9.5.2 Using Multiple Reports Server Clusters in Oracle Forms Services

If your Oracle Forms application from a prior release uses multiple Reports Server cluster names, you can map each of those cluster names to a different Reports Server. An Oracle Forms application that includes a Reports Server cluster name will fail to bind to the Reports Server cluster it references.

To resolve this issue, the reports_servermap element maps a cluster name to a Reports Server name. This avoids the necessity to change the cluster name in all Oracle Forms applications.

An Oracle Forms application can call Oracle Reports in the following ways:

- Using `RUN_REPORT_OBJECT`: If the call specifies a Reports Server cluster name instead of a Reports Server name, the `reports_servermap` environment variable must be set in the Oracle Forms Services `default.env` file. If your Oracle Forms application uses multiple Reports Server cluster names, you can map each of those cluster names to a different Reports Server using reports_servermap in rwservlet.properties, as follows:

  ```
  <reports_servermap>

  cluster1:repserver1;cluster2:repserver2;cluster3:repserver3

  </reports_servermap>
  ```

  For example, if your Oracle Forms application includes 3 clusters with names `dev_cluster`, `prd_cluster`, and `qa_cluster` in 10.1.2, you can map these cluster names to respective server names in later releases, as follows:

  ```
  <reports_servermap>

  dev_cluster:dev_server;prd_cluster:prd_server;qa_cluster:qa_
  server

  </reports_servermap>
  ```

  For more information about using RUN_REPORT_OBJECT against a Reports Server cluster in 11g, see My Oracle Support at For more information about calling Reports from Forms with `RUN_REPORT_OBJECT`, see My Oracle Support note 1074804.1 at `http://support.oracle.com`.

- Using `WEB.SHOW_DOCUMENT`: In this case, the request is submitted to rwservlet. If the call specifies a Reports Server cluster name instead of a Reports Server name, the reports_servermap element must be set in the `rwservlet.properties` file. For example:

  ```
  <reports_servermap>

  cluster:repserver

  </reports_servermap>
  ```

For more information, see *Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services*.

9.5.3 Integrating Forms and Reports Installed in Different Instances

In 11*g*, Forms and Reports can be configured separately in different instances. If you chose to install Forms and Reports in different Oracle instances, and later require Forms and Reports integration, you need to manually configure files required to establish communication with Reports Servers. For more information, see *Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services*.

9.6 Enabling and Configuring Proxy Users

This section contains the following:

9.6.1 Proxy User Overview

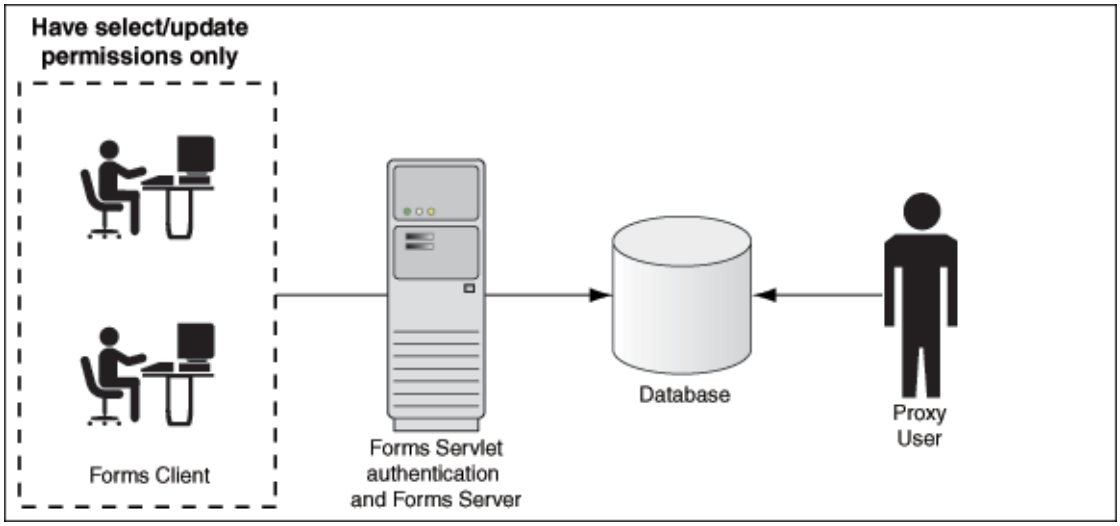
Many large applications, including Oracle's own E-Business Suite, use a single username for all connections. This makes it possible to manage users in a way that often suits large companies better but it creates a problem with auditing. All inserts, updates and removals of records appear, from the database's perspective, to have been done by a single user. To restore auditing, the application developers must write and implement customized auditing code in the database that requires a user name to be passed to the database from the application. This step not only takes development time, but also duplicates functionality that is already implemented in the Oracle Database.

The second issue is security. If that single user access is ever compromised, the compromised user will have access to the entire application schema.

To address these two issues, Oracle Database supports proxy user authentication, which allows a client user to connect to the database through an application server, as a proxy user.

Figure 9–3 describes the authentication of a Forms proxy user.

Figure 9–3 Proxy User Authentication



- Oracle Forms authenticates the user through Oracle Internet Directory or LDAP, as shown in the center of the image.
- Forms then connects as the proxy user with or without a password, passing in the real username from the Oracle Internet Directory repository.
- Typically, the proxy user is configured with least set of privileges. In the following procedure, the proxy user has "connect" and "create session" privileges.
- The database accepts the `create` session action for the proxy user and uses the real username in audits and access control.
- The Oracle Internet Directory user cannot connect to the database independently without configuration of the proxy user account.
- The proxy user account isolates the client from direct SQL*Plus connections.

9.6.2 Enabling Proxy User Connections

To use a proxy support in Forms, you first need to create a proxy user. In this example, the proxy user is called `midtier`:

1. Create a proxy user in the database.

```
SQL> CREATE USER midtier IDENTIFIED BY midtierPW;
```

2. Assign connect and create session privileges to midtier:

```
SQL> GRANT CONNECT,CREATE SESSION TO midtier;
```

   At this point, this proxy user has connect and create session privileges and has no grants on any of the user schemas.

3. Create a database user which has one-to-one mapping with a SSO username (that is, if `appuser` is the SSO username create database user `appuser`).

```
SQL> CREATE USER appuser IDENTIFIED BY appuserPW;
```

4. Assign create session privileges to appuser.

```
SQL> GRANT CREATE SESSION TO appuser;
```

5. To make it possible to connect through the midtier user you need to alter the database user:

   ```
   SQL> ALTER USER appuser GRANT CONNECT THROUGH midtier;
   ```

   The user `appuser` can now connect through the midtier account.

   Alternatively, you can define the roles that the proxy user can connect to the database as

   ```
   SQL> ALTER USER appuser GRANT CONNECT THROUGH midtier WITH ROLE <role_name>;
   ```

   Repeat Step 3 and 4 for all database users who need to use the proxy user account.

It is also possible to set up the database users in Oracle Internet Directory with the help of the database functionality called Enterprise User Security. If you choose this method, the proxy user is the only user defined in the database and the additional benefit of easy administration is gained. For more information on using Enterprise User Security, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory 11g Release 1 (11.1.1)*.

The application user's password is not presented to the database; only the user name and the proxy user's user name and password. Forms, with the help of OCI calls, issues the equivalent of:

```
SQL> connect midtier[appuser]/midtierPW@databaseTnsName
```

For example, suppose your application always connects to the database using midtier. This midtier now informs the database that the actual user is `appuser`. Without using proxy users, the SQL command `select USER from DUAL` would return midtier, but, using proxy users, this query returns `appuser`. This essentially tells the database to trust that the user is authenticated elsewhere and to let the user connect without a password and to grant the connect role.

Note:

- In the Step 3 of the above procedure, the database users are typically configured to have a subset of permissions granted to a schema. For example, appuser is granted `CREATE` permissions to the schema `app_schema` with the SQL command:

  ```
  SQL> GRANT CREATE ON SCHEMA app_schema TO appuser
  ```

  Thus, the appuser is restricted to perform only a set of actions in proxy user mode.

- When the database user (for example, appuser) is connected in proxy mode, user actions of the database users are audited rather than that of the proxy user. For more information on user action auditing, refer to the Oracle Database documentation at `http://www.oracle.com/technology/documentation/index.html`.

9.6.3 Enabling SSO in formsweb.cfg

Create a configuration section in `formweb.cfg` for single sign-on (for example, `ssoapp`) and set `SSOProxyConnect` to `yes` and `ssoMode` to `true`.

The username and password that is used for the proxy connection is defined in the RAD entry in Oracle Internet Directory for the user that is logging on. If `ssoProxyConnect=yes`, the connect string equivalent issued by Forms is in effect:

```
SQL> connect RADUsername[appuserName]/RADPassword@databaseTnsName
```

9.6.4 Accessing the Forms Application

After enabling proxy user connections and single sign-on, perform the following steps to access the forms applications:

1. Run the forms application with the URL `http://<host name>:<http port>/forms/frmservlet?config=ssoapp` where `ssoapp` is the name of the configuration section with single sign-on (`ssoMode`) is enabled.
2. Use the single sign-on user name and password to log in (in this example given in Section 9.6.2, "Enabling Proxy User Connections", the single sign-on username is `appuser` and password is `appuserPW`).

9.6.5 Changes in Forms Built-ins

The Built-in `get_application_property` now takes a new parameter called `IS_PROXY_CONNECTION` (a Boolean). When this parameter is supplied, the call returns `true` if the form is running in proxy user mode, `false` otherwise.

9.6.6 Reports Integration with Proxy Users

The integration with Reports is maintained when a proxy user is used in Forms. The Oracle Reports administrator has to set up a proxy user. Ensure that the following configuration has been completed in the Reports configuration files.

In `rwserver.conf`, enter the Forms configuration section name (`frm_config_name`) and database SID name that is configured for proxy user support (dbname).

```
<dbProxyConnKeys>

    <dbProxyKey name="frm_config_name"  database="dbname"/>

</dbProxyConnKeys>
```

In `rwservlet.properties`, ensure that Proxy mode is enabled.

```
<enabledbproxy>yes</enabledbproxy>
```

For more information about Reports configuration files, see the *Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services*

9.7 Configuring Oracle Internet Directory

The users connecting through a Forms application as proxy users must also be defined in OracleAS Single Sign-On Server and Oracle Internet Directory. Oracle Forms authenticates the user via OracleAS Single Sign-On Server (using OracleAS Single Sign-On Server with Forms is a requirement when employing a proxy user). Oracle Forms then connects to the database as the proxy user with a username and password that is in the RAD for the Oracle Internet Directory entry for the application user.

For more information on Oracle Forms and Identity Management integration, see Section 11.1.4, "Leveraging Oracle Identity Management Infrastructure."

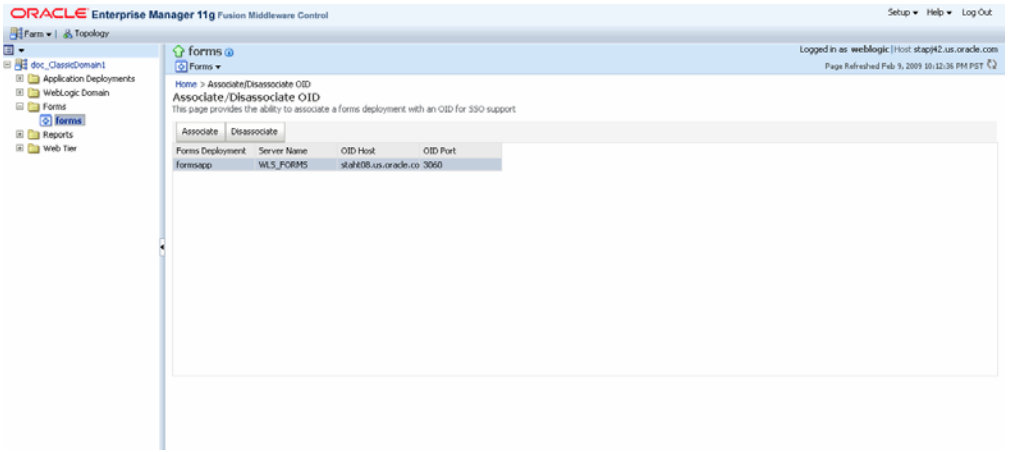
Note: When you change the Oracle Web Cache port using Enterprise Manager, regenerate the `osso.conf` and copy the generated `osso.conf` file to `$ORACLE_INSTANCE/config/OHS/<OHS_INSTANCE>/moduleconf` directory. Restart the Oracle HTTP Server and Oracle Web Cache for the changes to take effect.

To access the Associate/Disassociate OID page:

1. Start Enterprise Manager.
2. Navigate to the Forms **Home** page.
3. From the Forms menu, select **Associate/Disassociate OID**.

   The **Associate/Disassociate OID** page is displayed.

Figure 9–4 Associate/Disassociate OID



To Associate OID Host with a Forms Application

1. To associate an Oracle Internet Directory host with a Forms application for the first time, from the **Associate/Disassociate OID** page, select the Forms application. Click **Associate**.

   The Associate dialog appears.
2. Enter the Oracle Internet Directory Host details as described in Table 9–1, " Oracle Internet Directory Host Details".
3. Click **Associate**.

   The **Associate/Disassociate OID** page reappears.

Table 9–1 Oracle Internet Directory Host Details

Parameter	Description
OID Host	Select the Oracle Internet Directory Host from the list or select New OID host to add new Host details.
New OID host	Host name of the LDAP directory server. This field is enabled if you have selected to add new Oracle Internet Directory Host.
New OID Port	Port number on which LDAP is listening. This field is enabled if you have selected to add new Oracle Internet Directory Host.
Username	Oracle Administrator username

Table 9–1 (Cont.) Oracle Internet Directory Host Details

Parameter	Description
Password	Oracle Administrator password
Use SSL Port	Select this box if the connection to the Oracle Internet Directory Host should use SSL (in which case the port number provided should be the SSL port).

4. On the OracleAS Single Sign-On Server, run the `ssoreg.sh` script from `$ORACLE_HOME/sso/bin`.

```
ORACLE_HOME/sso/bin/ssoreg.sh
-oracle_home_path <ORACLE_HOME>
-site_name www.example.com
-config_mod_osso TRUE
-mod_osso_url http://www.oidtierexample.com:7777
-config_file osso.conf
-remote_midtier
```

   On Windows, run the `ssoreg.bat` file.

5. Copy the generated osso.conf file to `$ORACLE_INSTANCE/config/OHS/<OHS_INSTANCE>/`. For more information, see the *Oracle Application Server Single Sign-On Administrator's Guide* on OTN.

6. Restart the Oracle WebLogic Managed Server and the front-end OHS for the changes to take effect.

   To prevent users from being inadvertently disconnected from active forms sessions, ensure you choose to restart Oracle WebLogic Managed Server and the front-end OHS at a convenient time when users are not running any forms sessions.

To Disassociate OID Host from a Forms Application

1. From the **Associate/Disassociate OID** page, select the Forms application. Click **Disassociate**.

   A confirmation box appears.

2. Click **Yes**.

   The Oracle Internet Directory host is disassociated from the Forms application.

3. Restart the Oracle WebLogic Managed Server and the front-end OHS for the changes to take effect.

   To prevent users from being inadvertently disconnected from active forms sessions, ensure you choose to restart Oracle WebLogic Managed Server and the front-end OHS at a convenient time when users are not running any forms sessions.

To re-associate an OID Host with a Forms Application

1. From the **Associate/Disassociate OID** page, select the Forms application. Click **Disassociate**.

2. From the **Associate/Disassociate OID** page, select the Forms application. Click **Associate**.

   Enter the Oracle Internet Directory Host details as described in Table 9–1, " Oracle Internet Directory Host Details".

3. On the OracleAS Single Sign-On Server, run the `ssoreg.sh` script from `$ORACLE_HOME/sso/bin`.

```
ORACLE_HOME/sso/bin/ssoreg.sh
-oracle_home_path <ORACLE_HOME>
-site_name www.example.com
-config_mod_osso TRUE
-mod_osso_url http://www.oidtierexample.com:7777
-config_file osso.conf
-remote_midtier
```

   On Windows, run the `ssoreg.bat` file.

4. Copy the generated osso.conf file to `$ORACLE_INSTANCE/config/OHS/<OHS_INSTANCE>/`. For more information, see the *Oracle Application Server Single Sign-On Administrator's Guide* on OTN.

5. Restart the Oracle WebLogic Managed Server and the front-end OHS for the changes to take effect.

   To prevent users from being inadvertently disconnected from active forms sessions, ensure you choose to restart Oracle WebLogic Managed Server and the front-end OHS at a convenient time when users are not running any forms sessions.

10

Configuring and Managing Java Virtual Machines

This chapter contains the following sections:

10.1 Why Use Java Virtual Machine Pooling?

When a Forms application calls out to Java, a JVM is attached to each Forms process the first time the process makes a call. This JVM remains attached to each process for the remainder of the processes' lives, even though any individual process may never call out to Java again, potentially causing resource contention. JVM pooling makes provisions for sharing a limited number of JVMs among all participating Forms processes. Even though all Forms processes might at one point call out to Java, if only a subset of these call out to Java at any given point in time, only as many JVMs as are necessary at peak usage, need be started. Using JVM pooling brings the potential to significantly reduce resource usage for a Forms installation that calls out to Java.

When a Forms runtime process needs to execute Java, it sends a message to the Java Virtual Machine (JVM) that is contained in the JVM controller. The JVM creates a new thread for that Forms runtime process. The JVM then continues to listen for the next new request from a different Forms runtime process while the newly created thread processes the request and sends the results back to the Forms runtime process. For the life of this Forms session, the Forms runtime process communicates directly with that thread.

Java Virtual Machine pooling is a separate process that contains the JVM controller. With JVM pooling, the JVM runs outside of the Forms runtime process. The JVM can

also be shared by multiple Forms runtime processes. The JVM controller process is not a JVM itself, but a container that contains a JVM in a similar way that the Forms Runtime process contains an in-process JVM. Using JVM pooling is optional. Administrators can choose to not use JVM pooling and have the JVM contained in the Forms runtime process.

Java Virtual Machine (JVM) pooling works in conjunction with the Java Importer. It also works with Forms' ability to call out to Reports. The Java Importer allows developers at design time to reference Java classes from PL/SQL within the Forms Builder. At runtime, Forms uses a Java Virtual Machine (JVM) to execute Java code. In earlier versions of Oracle Forms, each Forms session that used the Java Importer had its own JVM instance to execute Java code. In this model, each JVM consumes memory on the server, and if there are many concurrent users, the amount of memory consumed by the multiple JVM processes becomes significant.

For more information on the Java Importer, see the Oracle Forms Developer online help.

When you enable JVM pooling, administrators can consolidate the number of running JVM instances so that the Forms sessions can share JVMs rather than each one having its own instance. The result is a large reduction in memory consumption, thus freeing up more resources on your server.

You also need to consider JVM pooling in application design and deployment. For more information, see Chapter 10.5, "Design-time Considerations".

10.1.1 JVM Pooling in Forms and Reports Integration

In 10*g*, Forms Runtime process creates a separate JVM before calling Reports and Reports uses this JVM to execute the java methods. This JVM is part of the Forms Runtime process. In 10*g*, the JVM pooling feature is used only by the Java Importer. However, in 11*g*, with JVM pooling enabled, Oracle Forms Services uses a shared JVM controller for Oracle Reports requests.

Instead of each Forms Runtime process having its own instance of the JVM, JVMs can be shared by multiple Forms Runtime processes. With JVM pooling, a process called JVM controller is available which houses the JVM. Forms Runtime processes can share this JVM. This would result in a large reduction of memory consumption, freeing more resources on the server.

A form can be configured to use a specific JVM controller using the jvmcontroller parameter. The jvmcontroller parameter indicates to the Forms Runtime process which JVM controller to use. This can be set in the Forms Configuration File, `formsweb.cfg`. Alternatively, this information can also be passed as a parameter in the URL for invoking the Forms Application. The parameters that need to be used during startup of the jvmcontroller have to be specified in the JVM controller's configuration file, `jvmcontrollers.cfg`.

For more information on using JVM pooling for Reports integration, see Section 10.10, "Integrating Forms and Reports".

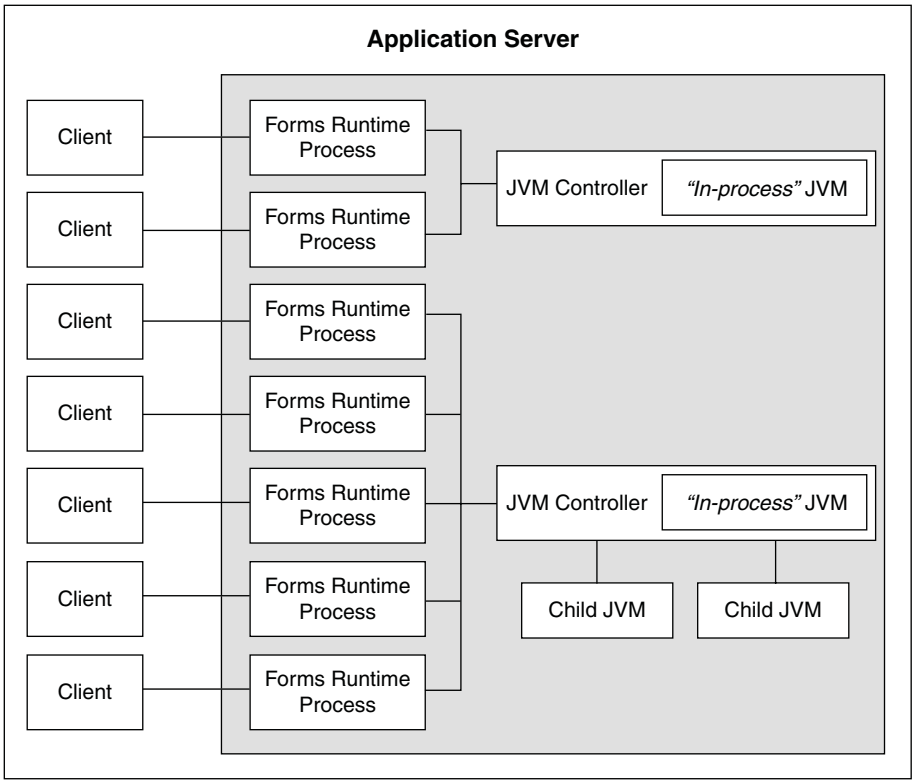
10.2 About Child Java Virtual Machine Processes

Since each Forms runtime process has its own thread within the JVM, there is concurrency. If the JVM reaches a specified number of concurrent requests, it will spawn a *child* JVM to share the load. Moreover, it's possible to have multiple JVM controllers, each of which may have multiple child JVMs.

For example, different Forms applications may want to use different JVMs with different options or classpaths. You can specify which JVM controller and Forms application should be used in the named sections of the Forms configuration file (`formsweb.cfg`). See Section 10.8.6, "Forms Configuration File Settings" for more information.

Figure 10–1 shows an example of what an environment might look like using JVM pooling. There are two JVM controllers: the first one is using only its in-process JVM, the second one is using three JVMs.

Figure 10–1 Multiple JVM Controllers with Child Processes



Although it's not shown in Figure 10–1, each JVM controller has a unique name which is used in starting and stopping, or for referencing in the Forms configuration file.

Figure 10–1 is conceptual only in that it shows different Forms applications using different JVM controllers. However, the Forms runtime process does not communicate with the JVM controller, but directly with one of the available JVMs. Therefore, the first two clients in the diagram can only use the in-process JVM; the rest have three available JVMs to work with.

When the performance of a JVM degrades significantly, it probably means it is servicing too many requests. In that case, it is possible to have multiple "child" JVMs for the same JVM controller which get created dynamically as needed.

The JVM parameter `maxsessions` specifies how many Forms runtime processes are allowed to attach to a JVM before a new child JVM is created. When a child JVM is started, it inherits the same parameters as the JVM controller.

If any JVM has `maxsessions` connections, it does not take any request from new Forms runtime processes. When a new Forms runtime process first attempts to execute Java code, it attaches to a JVM that is available, that is, has fewer than `maxsessions`

connections. The method of choosing the JVM is entirely arbitrary; there is no load balancing or round-robin algorithm.

If a JVM reaches maxsessions connections, but another JVM has not, no new JVM is created. If all JVMs have simultaneously reached maxsessions connections, another child JVM is created, and so on.

Child JVMs are not automatically removed when the load is reduced. So if you want to remove some child JVMs, the JVM controller must be stopped, which also stops all child JVMs. Then the JVM controller can be restarted.

The scope of a child JVM is within the context of a JVM controller namespace. For example, if you have two JVM controllers, `ordersJVM` and `hrJVM`, ordersJVM and its child JVMs do not affect – nor are affected by – `hrJVM` or its child JVMs.

10.2.1 Child JVM Example

Suppose the JVM controller called ordersJVM has `maxsessions=50`. Each Orders application that runs sends requests to ordersJVM. Each time a new Forms runtime process sends a request to ordersJVM, a new thread is created that communicates with the Forms runtime process. The JVM controller then returns to listening for new requests. As users end their sessions, the threads in the JVM are also terminated.

When the ordersJVM controller receives the 50th concurrent request (not necessarily the first 50 users because some of them may have quit before the later users started) it will spawn a child JVM. Since it inherits its parent's settings, `maxsessions` for this child JVM will also be `50`. At this stage, the JVM controller has 50 connections, and the child JVM has none.

As new users start this Oracle Forms application and execute Java code, the Forms runtime process attaches to a JVM that is listening within the JVM controller namespace. Since the JVM controller has 50 connections, it is unavailable and the child JVM receives the request. Later, when the parent JVM controller has fewer connections because some users have quit their applications, it is available to receive new requests as long as it has not reached `maxsessions` connections.

While all this is going on, the `hrJVM` is operating independently. Overflow connections from `ordersJVM` will not connect to `hrJVM`, only to child JVMs of `ordersJVM`.

10.3 About Multiple JVM Controllers

The JVM pooling architecture allows you to have multiple JVM controllers, each of which may have child JVMs. You would use multiple JVM controllers if:

- You want each application to have its own JVM controller so that it can be started and stopped independently of others.
- Different applications require different settings. For example, you may not want to mix classpaths or JVM settings between different controllers.
- You want to monitor resource usage of the JVM controllers from Fusion Middleware Control. If different JVM controllers are used by different applications and/or groups of users, you can determine how resources are being consumed by your Java Importer code.
- You have multiple development, test, or production environments on the same computer.
- You do not want different applications to share static data.

10.4 JVM Pooling Usage Examples

Consider, for example, an Oracle Forms application that has a user interface button. When a user presses the button, Oracle Forms takes the value from a field on the screen, and passes it to Java (using the Java Importer feature) to do some complex calculation which cannot be done in PL/SQL. The result is then returned and displayed in a field in the Form. One JVM process is running to execute this Forms session.

Figure 10–2 shows how this Oracle Forms session has its own **in-process JVM** because JVM pooling is not enabled. In the left side of the image, there are multiple clients running their own Forms session. In the center of the image, each client makes a call to its own Forms Runtime process, which contains its own JVM process.

Figure 10–2 Forms Runtime with no JVM Pooling

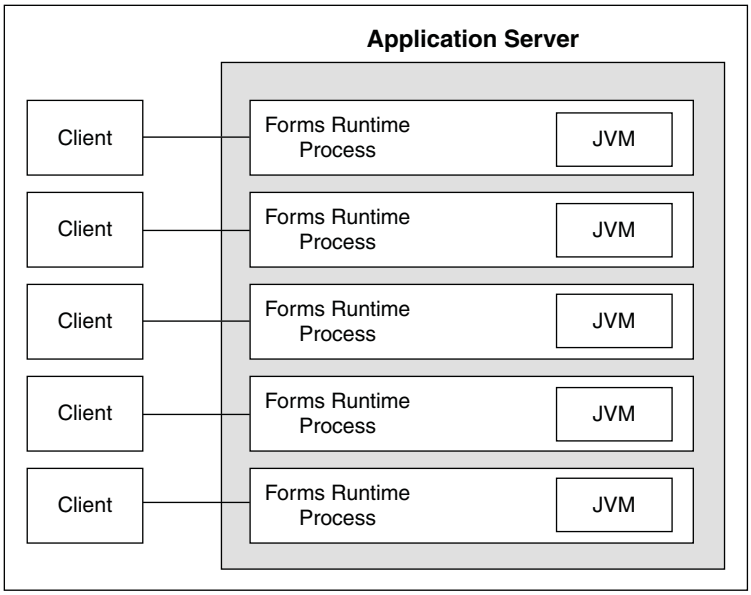
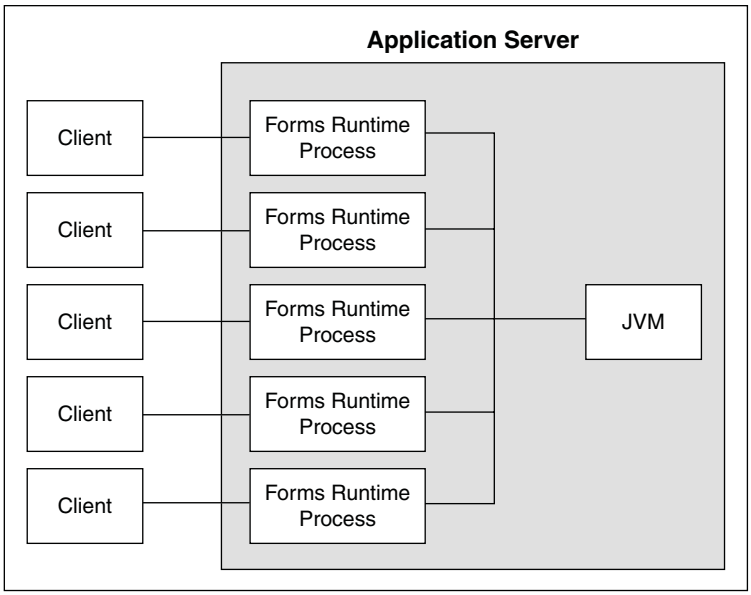


Figure 10–3 shows the Forms Runtime processes sharing a single JVM process when JVM pooling is enabled, as shown in the right side of the image.

Figure 10–3 Forms Runtime with JVM Pooling Enabled



In this example, five clients working in the same application through their own runtime processes are using a pooled JVM process instead of each Forms Runtime process spawning its own JVM instance. This can be a significant savings in memory usage and system resources.

10.5 Design-time Considerations

This section contains the following:

- Section 10.5.1, "Re-importing Your Java Code"
- Section 10.5.2, "About Sharing Static Variables Across Multiple JVMs"

10.5.1 Re-importing Your Java Code

If you used the Java Importer feature of Oracle Forms prior to the availability of JVM Pooling, you will need to reimport your Java classes before using JVM pooling. When you originally imported your Java classes, PL/SQL wrappers for the Java classes were generated, which you can see in the Program Units that were created in your Form. However, the PL/SQL wrappers that are generated by the Java Importer to utilize JVM pooling are different.

From Oracle Forms Services 10*g* and later, the Java Importer generates the "new" PL/SQL wrappers. If you want to use the Java Importer, but do not wish to take advantage of JVM pooling, the in-process JVM will work with the new PL/SQL wrappers. It will also continue to work with the older-style PL/SQL wrappers.

10.5.2 About Sharing Static Variables Across Multiple JVMs

One advantage of JVM pooling is the ability to share data between instances of a class by using static variables. However, static variables will be shared between instances of the same class within a JVM, but not across JVMs. You will need to plan accordingly.

For example, suppose your loan class has a static variable called `interestRate` because all instances use the same interest rate in calculations. If you are using only

one JVM, and one of the instances of your loan class changes `interestRate`, all of the other instances will be affected (which is what you want).

However, if the JVM controller has one or more child JVMs, there may be at least two JVMs. If interestRate changes in one JVM, the loan instances in the other JVMs won't see this new value. For more information about managing child JVMs, see Section 10.2, "About Child Java Virtual Machine Processes". Prior to JVM pooling, if you changed interestRate it would not affect any other instances because each Oracle Forms Runtime process had its own in-process JVM.

If you rely on static variables to share information between instances of your class, ensure that no child JVM is spawned by setting `maxsessions` to `65535`.

10.6 Overview of JVM Configuration

To configure JVM using Fusion Middleware Control, perform the following steps:

1. Using Fusion Middleware Control, add a new configuration section or modify an existing section in `formsweb.cfg` to enable or disable use of JVM controller for applications. For more information, refer to Section 10.8.6, "Forms Configuration File Settings".
2. Ensure CLASSPATH is updated in `default.env` or in `jvmcontrollers.cfg`.
3. Using Fusion Middleware Control, configure the JVM parameters. For more information, refer to Section 10.8.3, "Managing Parameters".
4. Start the JVM controller. For more information, refer to Section 10.8.5, "Starting and Stopping JVM Controllers with Fusion Middleware Control".

10.7 Managing JVM Controllers from the Command Line

If you manage JVM controllers from the command line, you must know the options to start and stop them, as well as specify the environment. You can only access the JVM controllers on the same computer from which they are running.

Note: The mechanics for controlling the JVM controller as described in this chapter are mostly relevant at the command line. It is easier to use Fusion Middleware Control with its user-friendly screens and online help. Fusion Middleware Control users are still urged to read through the following information, however, to understand what the different fields and options mean, and how the JVM controller works.

10.7.1 JVM Controller Command Examples

This section describes examples of JVM controller commands. For a detailed explanation on the example, see Section 10.8.7, "Startup Example."

- `dejvm -start jvmcontroller=hrJVM`

  Starts a JVM controller with ID hrJVM. The controller name hrJVM is defined as a named section in the configuration file. Therefore, JVM options and classpath parameters are taken from the configuration file. `maxsessions` is `50` as defined in the Default section, and other parameters take their default values.

- `dejvm -start jvmcontroller=myJVM`

  Starts a JVM controller with ID is myJVM. Since no option was specified, and there is no named section in `jvmcontrollers.cfg`, the JVM options parameter is

"-Xms512m -Xmx1024m" and maxsessions=50 as set in the Default section. The other parameters take on their default values. For instance, the CLASSPATH value is the system CLASSPATH.

- `dejvm -start jvmcontroller=hrJVM jvmoptions="-Xms128m -Xmx256m" maxsessions=75`

  Sets the classpath to /myJava/hrClasses as defined in the named section. JVM options are "-Xms128m -Xmx256m" because the command line overrides the jvmcontrollers.cfg file. Similarly, maxsessions is 75. All other parameters take on their default values.

- `dejvm -start jvmcontroller=myJVM maxsessions=100 classpath=/myJava/myClasses;/moreJava/moreClasses`

  The controller has jvmoptions="-Xms512m -Xmx1024m" as defined in the default section of jvmcontrollers.cfg. maxsessions is 100 which overrides the default section, and classpath is /myJava/myClasses;/moreJava/moreClasses. All other parameters take on their default values.

- `dejvm -stop jvmcontroller=hrJVM`

  Stops the hrJVM controller. It must already be started for you to issue this command successfully.

10.7.2 Command Restrictions

Keep these command restrictions in mind:

- The commands are case sensitive.
- You can only issue one command at a time to a JVM controller.
- You can only issue a command to one JVM controller at a time.

The available commands for the JVM controller (or the dejvm process) are specified in Table 10–1. If you are using Enterprise Manager, there are screens that have an interface for issuing these commands. If you are using the command line, you may not be able to manage the JVM controller using the Enterprise Manager.

10.7.3 Start Command Parameters

Table 10–1 describes the JVM parameters used to start the JVM from the command line.

Table 10–1 JVM Parameters

Parameter	Description
jvmcontroller	Enter a name for this JVM. This name must contain a legal Oracle identifier that starts with a letter and contains an alphanumeric character, '_', '$' or '#' . An Oracle identifier has a maximum length of 30 bytes. Hint: You may want to enter a name based on the application that will be accessing it. You cannot change the name of this JVM controller later.
maxsessions	Specifies the maximum number of concurrent Oracle Forms sessions this JVM will serve before a new JVM is spawned. This value will override any set for the default JVM controller.

Table 10–1 (Cont.) JVM Parameters

Parameter	Description
classpath	When you specify a classpath, it will override the system classpath or any classpath specified in your environment or any classpath set for the default JVM controller.
jvmoptions	Enter any valid options to pass to the JVM. This value will override any set for the default JVM controller. Refer to the Sun Java documentation for a list of valid JVM startup options.
logdir	Leave Log Directory blank to use the log location for the default JVM controller. If any other directory is set, the log file may not be accessible through Enterprise Manager.
logging	On, or Off.

10.8 Managing JVM Pooling from Fusion Middleware Control

Fusion Middleware Control provides a Web-based environment to manage all available JVM pooling options. It also lists all JVM controllers in your environment and allows you to (remotely) manage them. For example, you can start and stop JVM controllers; add new ones; or reconfigure existing ones. In addition, Fusion Middleware Control also provides metric information such as resources (memory and CPU) that are consumed by JVM controllers, number of Forms connected, total JVMs, and so on.

While the Forms runtime process interacts directly with a JVMs, the JVM controller manages the JVM, such as starting and stopping a JVM, or getting the state of one, etc. For example, when an administrator stops the JVM controller, the JVM controller ensures that all child JVMs are terminated. You use Fusion Middleware Control to manage the JVM controller.

The JVM controller can be started in three ways:

- From Fusion Middleware Control
- When a Forms application that is bound to an existing JVM controller requests that the controller start up
- From the command line

Fusion Middleware Control reads the JVM controller configuration file. It works in a similar way to the Forms configuration file (`formsweb.cfg`) in that it contains name-value pairs, has a default section, and has named sections. The parameters contained in `jvmcontrollers.cfg` correspond to the start parameters of the JVM controller.

> **Note:** You cannot change the location or name of the JVM controllers configuration file.

When you start a JVM controller, it takes its settings from the configuration file. You may specify none, some, or all options in this file, both in the default section and in named sections.

Use the JVM Configuration and JVM Controller pages in Fusion Middleware Control to manage JVM pooling tasks:

- Section 10.8.1, "Common Tasks in the JVM Configuration Page"
- Section 10.8.2, "Managing JVM Configuration Sections"

10.8.1 Common Tasks in the JVM Configuration Page

This section describes the common tasks that you can do to edit configuration with the sections of a JVM configuration file and their parameters.

Table 10–2 describes the tasks you can do with the configuration sections within a JVM configuration file:

Table 10–2 Tasks for Working with Configuration Sections

Task	Description	Comment
Create Like	Creates a copy of a configuration section.	Use to create a configuration section based on the parameters of an existing configuration section.
Edit	Opens the Edit Description dialog.	Allows editing the text description of a configuration section.
Delete	Opens the Confirmation dialog when deleting a configuration section.	Irrevocably deletes a configuration section and its contents when you press Delete in the Confirmation dialog.
Create	Opens the Create Section dialog.	Creates a new configuration section. You must supply a required name and an optional description for it.

Table 10–3 describes the tasks that you can do to modify the parameters within a named configuration section:

Table 10–3 Tasks for Working with Parameters in a Named Configuration Section

Task	Description	Comment
Revert	Allows you to revert back to the previous version of the configuration section.	Does not allow you to revert individual changes in a configuration section.
Apply	Applies and activates all changes made to parameters in a configuration section.	Once applied, you cannot revert changes to individual parameters.
Add	Opens the Add Parameter dialog.	Add a parameter to a configuration section based on a mandatory name and an optional value and description.

Table 10–3 (Cont.) Tasks for Working with Parameters in a Named Configuration Section

Task	Description	Comment
Delete	Deletes a parameter.	Use Apply to save changes or Revert to discard them. Once applied, you cannot revert changes to individual parameters.

10.8.2 Managing JVM Configuration Sections

This section describes creating, editing, duplicating, and deleting named JVM configuration sections.

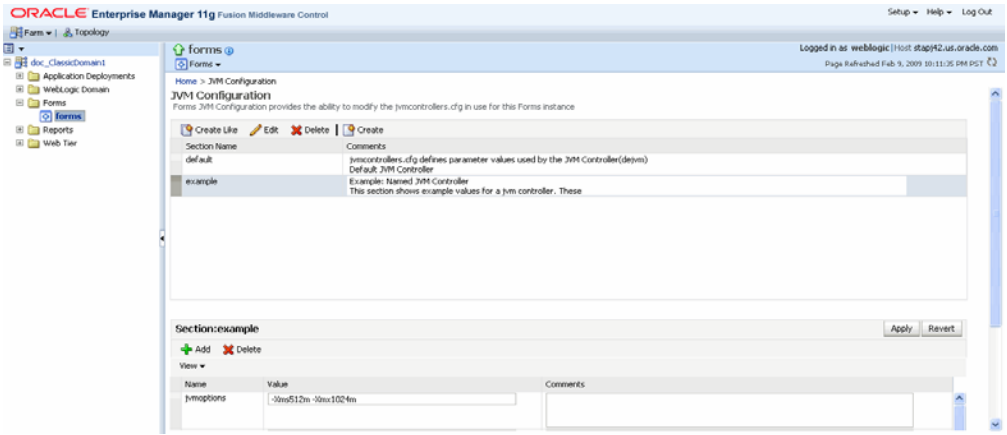
10.8.2.1 Accessing the JVM Configuration Page

To access the JVM configuration page:

1. Start the Enterprise Manager Fusion Middleware Control.
2. From the Fusion Middleware Control main page, click the link to the Forms Services instance that you want to configure.
3. From the Forms menu list, select the **JVM Configuration** menu item.

   The JVM Configuration page (Figure 10–4) is displayed.

Figure 10–4 JVM Configuration Page



10.8.2.2 Creating a New Configuration Section

You can create new configuration sections in `jvmcontrollers.cfg` from the **JVM Configuration** page of Fusion Middleware Control. These configurations can be requested in the end-user's query string of the URL that is used to run a form.

To create a new configuration section:

1. From the Fusion Middleware Control main page, click the link to the Forms Services instance that you want to configure.
2. From the Forms menu list, select **JVM Configuration**.
3. Click **Create**.

   The **Create** dialog appears.

4. Enter a name and description for your new configuration section and click **Create**.

   The new configuration section is added.

10.8.2.3 Editing a Named Configuration Description

You can edit the description (comments) for a named configuration from the **JVM Configuration** page.

To edit a named configuration description:

1. In the **JVM Configuration** region, select the row containing the named configuration for which you want to edit the description.
2. Click **Edit**.
3. The **Edit Description** dialog appears.
4. Enter the description in the Comments field.
5. Click **Save**.

   The **Edit Description** dialog box is dismissed, and your changes are saved and displayed.

10.8.2.4 Duplicating a Named Configuration

You can make a copy of a named configuration for backup purposes, or create new configuration sections from existing configuration sections.

To duplicate a named configuration:

1. In the **JVM Configuration** region, select **Create Like**.
2. In the Create Like dialog, from the **Section to Duplicate** menu, select the name of an existing configuration section you want to duplicate.
3. In the **New Section Name** field, enter a name for the new configuration section. The name for the new configuration section must be unique.
4. Click **Create**.

   A new section with exactly the same parameters, parameter values and comments of the section you are duplicating is created.

10.8.2.5 Deleting a Named Configuration

When you delete a named configuration section, you delete *all* the information within it. If you only want to delete specific parameters, see Section 10.8.3, "Managing Parameters".

To delete a named configuration:

1. From the **JVM Configuration** region, select the row of the configuration section you want to delete.
2. Click **Delete**.

   The **Confirmation** dialog appears.
3. Click **Delete**.

   The configuration section is deleted.

   Oracle Enterprise Manager returns to the **JVM Configuration** page and displays the remaining configurations.

Note: You cannot delete the Default configuration section.

10.8.3 Managing Parameters

Use Fusion Middleware Control to manage parameters within a named configuration. You can add, edit, or delete parameters using Fusion Middleware Control.

To edit a parameter in a configuration section:

1. From the **JVM Configuration** region, select the row of the configuration section that contains the parameter(s) you want to edit.
2. Select the row of the parameter you want to edit. Enter the Value and Comments.
3. Click **Apply** to save the changes or **Revert** to discard them.

To add a parameter to a configuration section:

1. In Fusion Middleware Control, from the **JVM Configuration** region, select the configuration section row for which you want to add a parameter.
2. Click Add to add a new parameter.

   The Add dialog box is displayed.
3. Enter the Name, Value and Comments for the parameter.
4. Click **Create** to add the parameter.
5. Click **Apply** to save the changes or **Revert** to discard them.

To delete a parameter in a configuration section:

1. In Fusion Middleware Control, from the **JVM Configuration** region, select the configuration section from which you want to delete a parameter.
2. Select the row that contains the parameter you want to delete.
3. Click **Delete**.
4. Click **Apply** to save the changes or **Revert** to discard them.

10.8.4 JVM Configuration Parameters and Default Values

Table 10–4 describes the JVM configuration parameters and their default values.

Table 10–4 JVM Configuration Parameters

Parameter	Description	Default Value
Maximum Sessions per JVM	Specifies the maximum number of concurrent Oracle Forms sessions the default JVM will serve before a new JVM is spawned.	65535
Classpath	When you specify a classpath, it will override the system classpath or any classpath specified in your environment.	`$ORACLE_HOME/jdk/bin/java`
JVM Options	Enter any valid options to pass to the JVM. Refer to the Sun Java documentation for a list of valid JVM startup parameters.	Null

Table 10–4 (Cont.) JVM Configuration Parameters

Parameter	Description	Default Value
Log Directory	Leave Log Directory blank to use the log location for the default JVM controller. If any other directory is set, the log file cannot be viewed through Enterprise Manager.	`$ORACLE_INSTANCE/FRComponent/frcommon/tools/jvm/log`
Logging	Specifies whether logging is enabled or not. Valid values: On, Off.	`On`
Comment	Add any comments about this default JVM in this text area.	Null

10.8.5 Starting and Stopping JVM Controllers with Fusion Middleware Control

Fusion Middleware Control is the recommended tool for managing Oracle Forms Services, such as starting, stopping, and restarting a JVM controller.

If a JVM controller is down, you can start it. If a JVM controller is already running, you can restart it without first having to manually stop it. Fusion Middleware Control does this step for you.

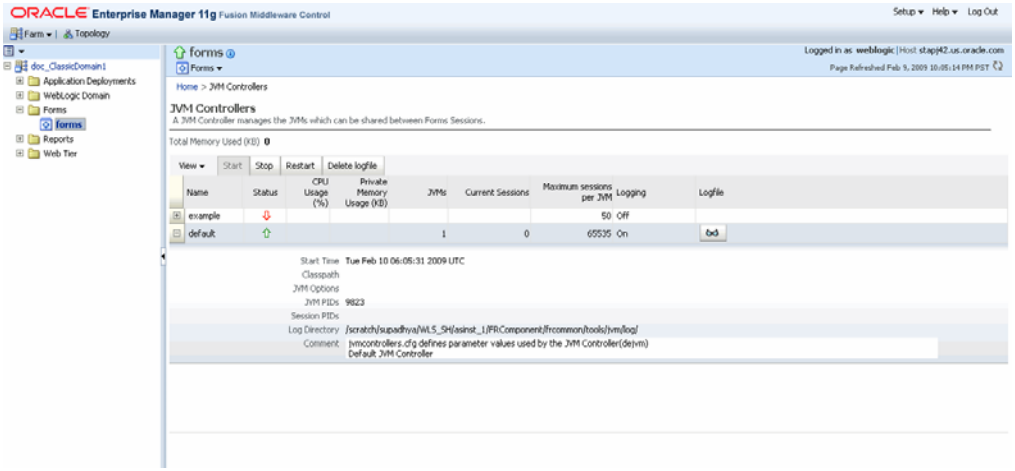
> **Note:** Ensure that users have stopped the forms sessions that are using the JVM controller before you stop or restart the JVM. Users may want to restart sessions when the JVM is restarted.

To access the JVM Controller page:

1. Start the Enterprise Manager Fusion Middleware Control.
2. From the Forms home page, select **JVM Controller**s.

   The **JVM Controllers** page (Figure 10–5) is displayed.

Figure 10–5 JVM Controller Page



To start a JVM controller that is not running:

1. From the Forms menu, select **JVM Controllers**.

   The **JVM Controllers** page is displayed.

2. Select the JVM controller that you want to start. A JVM that is not running is indicated by a red, down arrow.
3. Click **Start**.

   When the JVM controller has started, a green, up arrow (Figure 10–5) is displayed in the Status.

To restart a running JVM controller:

1. From the Forms menu, select **JVM Controllers**.

   The **JVM Controllers** page is displayed.
2. Select the JVM controller to be restarted.
3. Click **Restart**.
4. Click **Yes** on the Confirmation dialog.

   The **JVM Controller page** reappears.

   When the JVM controller has restarted, a green, up arrow is displayed in the Status.

To stop a JVM Controller

1. From the Forms menu, select **JVM Controllers**.

   The **JVM Controllers** page is displayed.
2. Select the running JVM controller that you want to stop, indicated by a green, up arrow.
3. Click **Stop**.
4. Click **Yes** on the Confirmation dialog.

   When the JVM controller has been stopped, a red, down arrow (Figure 10–5) is displayed in the Status.

To view additional details of a JVM Controller

1. From the Forms menu, select **JVM Controllers**.

   The **JVM Controllers** page is displayed.
2. Click the plus symbol next to the JVM controller. The row is expanded to display additional details (Figure 10–5) of the JVM controller.

10.8.6 Forms Configuration File Settings

This section describes the JVM pooling parameters that are used in the Forms configuration file (`formsweb.cfg`) to enable or disable use of JVM controller for applications. The parameter names are not case-sensitive. You can use Fusion Middleware Control to administer the Forms configuration file.

Table 10–5, " Oracle Forms JVM Controller Startup Parameters" describes the startup options that you specify in the `formsweb.cfg` file.

For more information on modifying the parameters in `formsweb.cfg`, see Section 4.2.4, "Managing Parameters".

Table 10–5 Oracle Forms JVM Controller Startup Parameters

Parameter	Description
`jvmcontroller`	Valid values: name of jvmcontroller. In addition, you can specify no JVM by leaving it blank. Default value: none **Note**: In order to specify this parameter in `formsweb.cfg`, you must first specify this parameter in `otherparams` in the form `jvmcontroller=%jvmcontroller%`. For more information on otherparams, see Table 4–13, " Advanced Configuration Parameters". This parameter can be set globally in the default section, or any application section can choose to override it. This tells the Forms runtime process which JVM controller to use. It corresponds to the `jvmcontroller` parameter for the dejvm executable. If `jvmcontroller` does not have a value (`jvmcontroller=`), then the Forms runtime process will start its own in-process JVM, which means that the Java Importer uses pre-10*g* behavior.
`allowJVMControllerAutoStart`	Valid values: `true`, `false` Default value: true This parameter enables Oracle Forms to run the JVM controller if Forms is configured to use the JVM controller which is not already running.

10.8.7 Startup Example

This example illustrates an environment of multiple JVMs for multiple applications.

As shown in Table 10–6, `formsweb.cfg` is configured with four configuration sections.

Table 10–6 Multiple JVMs for Multiple Applications

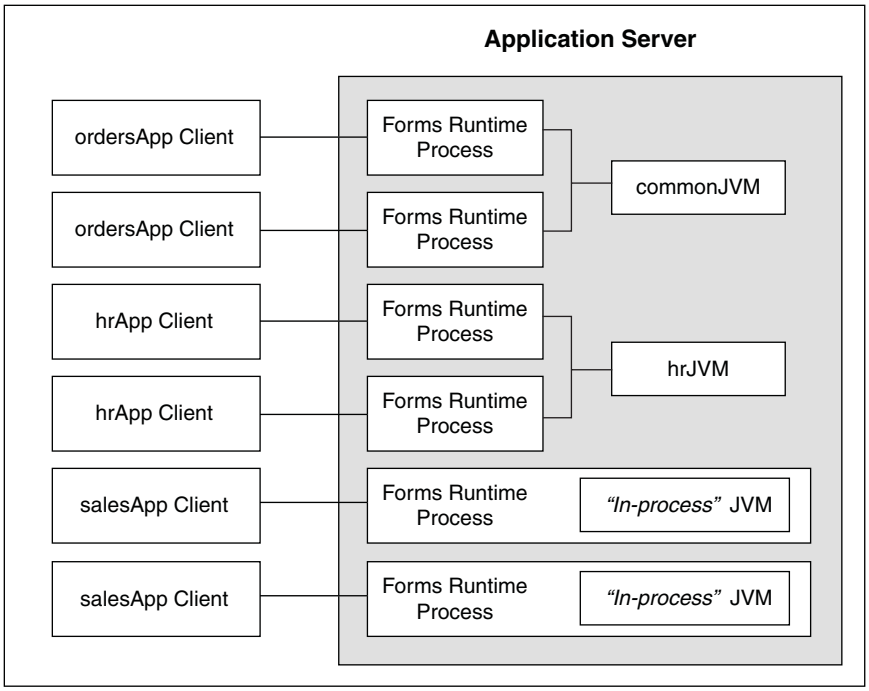
Named Configuration Section	JVM Configuration
`default`	`jvmcontroller=commonJVM`
ordersApp	None
hrApp	`jvmcontroller=hrJVM`
salesApp	`jvmcontroller=`

If a user starts an `ordersApp` application, and the application executes Java code, the Forms runtime process will route the request to the JVM controller named `commonJVM`. Because the `[ordersApp]` application section does not specify which JVM controller to use, the Forms runtime process uses the global one. If the JVM controller is not started, it will be dynamically started. If a second user starts the same application, it too will attach to `commonJVM`.

When a user starts an `hrApp` application and it executes Java code, the Forms runtime process sends the request to the JVM controller named `hrJVM` because the `[hrApp]` application section overrides the global setting. If the JVM controller is not started, it will be dynamically started. When a second user starts the same application, it too will attach to `hrJVM`.

When a user starts a `salesApp` application and it executes Java code, the Forms runtime process starts an in-process JVM in the same way the Java Importer works without JVM pooling. When a second user starts the same application, the application will get their own in-process JVM, thus consuming more memory, as shown in Figure 10–6:

Figure 10–6 Multiple JVMs for multiple applications



10.9 JVM Controller Logging

When logging is enabled, the JVM controller logs certain information to the log file:

- The values of the JVM parameters (maxsessions, classpath, and so on);
- When a JVM controller starts and stops;
- When a child JVM is spawned;
- When an Forms runtime process starts a new connection, along with its process ID

  This is useful for knowing which Forms runtime processes are connected to which JVM controller for diagnostics or administration;
- When an Forms runtime process session ends and disconnects from the JVM.

This section contains the following:

- Section 10.9.1, "Specifying JVM Default Logging Properties"
- Section 10.9.2, "Specifying the JVM Log Directory Location"
- Section 10.9.3, "Accessing Log Files"
- Section 10.9.4, "Deleting a Log File for a JVM Controller"

10.9.1 Specifying JVM Default Logging Properties

Use Fusion Middleware Control to manage the properties for JVM controller logging.

1. In the **JVM Configuration** page, select the the JVM configuration section.
2. For the Logging parameter, enter **On** or **Off**.
3. Click **Apply**.

10.9.2 Specifying the JVM Log Directory Location

You can specify the log file directory in the JVM controller. You can also specify the default JVM controller log file location for other JVM controllers to use.

To specify the log file directory location:

1. Create a JVM controller. For more information, see Section 10.8.2.2, "Creating a New Configuration Section" or Section 10.8.2.4, "Duplicating a Named Configuration".
2. Add the **Log Directory** parameter. For more information, see Section 10.8.3, "Managing Parameters."

   If you have duplicated a named configuration section that has **Log Directory** parameter defined in it, you can edit the existing parameter as given in the Section 10.8.3, "Managing Parameters."
3. Click **Apply** to save the changes.

   The **JVM Configuration** page reappears.

10.9.3 Accessing Log Files

When the log file exists, an icon is displayed in the Logfile column.

To access a log file:

- Click the Log File link in the Logfile column that is available for that JVM controller.

  The Log File page appears and displays the log information.

10.9.4 Deleting a Log File for a JVM Controller

Use Fusion Middleware Control to delete log files.

To delete a log file for a JVM controller:

1. From the **JVM Controllers** page, select the the target JVM.
2. Click **Delete Logfile**.

   The Delete Confirmation dialog appears.
3. Click **Delete**.

   The logfile is deleted and the **JVM Controllers** page reappears.

Note: If you delete a log file of a JVM that is running, the log file will be available again when the JVM is restarted. Logging is possible only when the JVM is restarted.

10.10 Integrating Forms and Reports

JVM Controller (`dejvm`) is used for Reports integration in Forms. All requests related to Reports such as running a report on Reports Server, getting the status of a Report, getting Reports output, or cancelling the job submitted to Reports Server are routed to the dejvm for dejvm-enabled runform to make calls to Reports.

To use `dejvm` for reports integration, perform the following steps. These settings are not required when Oracle Forms makes Reports call directly.

To use dejvm for Reports integration:

1. Enable JVM pooling in `formsweb.cfg`. For more information, see Section 10.8.6, "Forms Configuration File Settings".
2. Two additional `.jar` files are required by `dejvm` for Reports integration. Set the classpath in `jvmcontrollers.cfg` to include these jars: `zrcclient.jar` (`$ORACLE_HOME/jlib/zrclient.jar`) and `rwrun.jar` (`$ORACLE_HOME/reports/jlib/rwrun.jar`).

> **Note:** If you want the Reports Server name to be returned when running a report using JVM controller, then the REPORTS_SERVERMAP environment variable must be defined in jvmcontrollers.cfg as shown below:
>
> ```
> [myjvm]
> jvmoptions=-DREPORTS_SERVERMAP=<value> <other-jvmoption-parameters>
> ```

10.11 JVM Pooling Error Messages

PDE-JM001: Unable to communicate with the JVM Controller: <jvm_name>.

Cause: Failed to start the JVM controller or connect to an existing JVM controller.

Action: Notify your administrator.

11

Forms Services Security Overview

The ability to control user access to Web content and to protect your site against people breaking into your system is critical. This chapter describes the architecture and configuration of security for Oracle Forms Services:

- Section 11.1, "Forms Services Single Sign-On"
- Section 11.2, "Configuring Oracle Forms Services Security"

> **See Also:** For additional information about security, refer to the following documents:
>
> - The *Oracle Fusion Middleware Security Overview* provides an overview of Oracle Fusion Middleware security and its core functionality.
> - The *Oracle Fusion Middleware Getting Started with Oracle Identity Management* provides guidance for administrators of the Oracle security infrastructure.

11.1 Forms Services Single Sign-On

Single Sign-on in Oracle Forms Services is available through `mod_osso`, an Oracle module for the Oracle HTTP Server. `mod_osso` authenticates a user against Oracle Single Sign-On Server, which in turn uses Oracle Internet Directory as a user repository, before further passing the Forms application request to the Forms servlet.

Forms applications expect a database connect string to be passed along with the application request, otherwise a logon dialog is shown. To retrieve the database connect information in a Oracle Single Sign-On Server environment, the Forms servlet queries Oracle Internet Directory for the value of the combined unique key that is constructed from the user's Oracle Single Sign-On Server name, the authenticated user name, and the name of the application that the user is requesting to start.

Resource Access Descriptors (RAD) are entries in Oracle Internet Directory that are defined for each user and application which contain the required database connect information. The Forms servlet reads the database connect information from the RAD and passes it along with the command line that starts the Forms Web application. Although the Forms authentication is still database-centric, `mod_osso` and the Forms servlet are now integrated in a Web-based Oracle Single Sign-On Server environment.

11.1.1 Classes of Users and Their Privileges

Historically, Forms applications use the database to authenticate application users. To use Oracle Forms Services with Oracle Single Sign-On Server, the user account and its connect information must be available in Oracle Internet Directory. Oracle Internet

Directory provides several ways of provisioning user data, using PL/SQL, Java or the Oracle Delegated Administration Services. Oracle Delegated Administration Services is a Web-based user interface for Oracle Single Sign-On Server users and delegated administrators to administer self-service data in Oracle Internet Directory for which they are authorized.

Once a user account is created in Oracle Internet Directory, the Resource Access Descriptors (RAD) entries can be created dynamically the first time that a user requests a Forms application, assuming the user knows about the database connect information required for this application.

Another option is to use the RAD entries that can be created using Oracle Delegated Administration Services. The default RAD entries are accessible for all users that are authenticated through Oracle Single Sign-On Server. Use the default RAD if all users share the same database connect information when running a particular Forms application on the Web. This way, users are authenticated individually by their Oracle Single Sign-On Server credentials; however, all users share a common database connect (information) for the application defined by a default RAD entry.

11.1.1.1 Default Single Sign-On Behavior for User Accounts

By default, OracleAS Single Sign-On Server is enabled and no proxy user is involved. Oracle Forms users need to authenticate with OracleAS Single Sign-On Server, retrieve Resource Access Descriptors from the identity store (which is usually Oracle Internet Directory) and use these credentials to connect to the database.

11.1.1.2 Users Using Database Proxy Functionality

There is a new Oracle Single Sign-On Server parameter, `ssoProxyConnect`. Setting this to `true` allows users to connect as proxy users. The user is then required to authenticate with Oracle Single Sign-On Server, and a Resource Access Descriptor is configured which holds the proxy user's username and password. There is additional database configuration that needs to be implemented by the database administrator to allow for proxy connections.

11.1.2 Resources That Are Protected

When you enable Oracle Single Sign-On Server for your Forms applications, you can secure your Forms applications with these features:

11.1.2.1 Dynamic Directives

The dynamic `mod_osso` directive runs Oracle Single Sign-On Server protected Forms applications. This directive can optionally be used to run non-protected Forms applications from the same Oracle Forms Services instance. These applications use the same configuration files and Forms servlet. Single sign-on is enabled for applications by a OracleAS Single Sign-On Server parameter in the application definition of the `formsweb.cfg` configuration file.

11.1.2.2 Dynamic Resource Creation in Oracle Internet Directory

In some previous releases of Oracle Forms Services, if no resource access descriptor (RAD) definition was found for a specific application and user, an error message was displayed which locked out the user from running that Forms application, despite having authentication to do so. In this release of Oracle Forms Services, you can now configure Oracle Forms Services to allow users to create the RAD for this application on the fly if it does not exist. The funtionality to redirect to DAS pages is achieved with the single sign-on parameter `ssoDynamicResourceCreate`.

11.1.2.3 Database Password Expiration when Using Single Sign-On

In some previous releases of Oracle Forms Services, the RAD information in Oracle Internet Directory was not updated if the database password had expired, and users then renewed them when connecting to a Forms application. In this release, Oracle Forms Services automatically updates the RAD information in Oracle Internet Directory whenever a database password is updated through Forms. There is no extra configuration necessary to enable this feature in Oracle Forms Services.

11.1.3 Authentication and Access Enforcement

For detailed information about the authentication flow of Oracle Single Sign-On Server support in Oracle Forms Services, such as when the first time the user requests an Oracle Forms Services URL, or from a partner application, see Section 9.1.1, "Authentication Flow".

11.1.4 Leveraging Oracle Identity Management Infrastructure

Oracle Forms Services has tighter integration with Oracle Internet Directory with minimal configuration. When you configure Oracle Single Sign-On Server Server for your Forms applications, Oracle Forms Services handles much of the configuration and interaction with Oracle Internet Directory.

With the absence of Repository API in 11*g*, Oracle Forms and Identity Management integration involves the registration of Forms application identity at the time of deployment when a relationship is established between Forms and the Oracle Internet Directory (OID) host. This process is know as associating with the Oracle Internet Directory. Related information such as Forms Distinguished Name (formsDN) and the password are stored in Credential Storage Framework (CSF). At run time, a JNDI connection is made to Oracle Internet Directory after extracting the required information from CSF. Oracle Forms and Identity Management integration involves the following:

- Integration at bootstrap: The Forms application entity (and Distinguished Name) with a password is created in Oracle Internet Directory.
- Integration at run time: Previously, the connection to Oracle Internet Directory used Repository API. In 11*g*, a JNDI call is used to directly connect to Oracle Internet Directory.

For more information about associating and disassociating Oracle Internet Directory, see Section 9.7, "Configuring Oracle Internet Directory."

11.2 Configuring Oracle Forms Services Security

Configuring security for Oracle Forms Services is done through Oracle Fusion Middleware Control. Online help is available for each screen. For more information, see Chapter 4, "Configuring and Managing Forms Services" and Chapter 9, "Using Forms Services with Oracle Single Sign-On".

11.2.1 Configuring Oracle Identity Management Options for Oracle Forms

Oracle Forms Services can be configured to create resources dynamically in Oracle Internet Directory, or have a user with no Oracle Internet Directory resource use a common resource.

For more information, see Chapter 9, "Using Forms Services with Oracle Single Sign-On".

11.2.2 Configuring Oracle Forms Options for Oracle Fusion Middleware Security Framework

For more detailed information about configuring and securing Oracle Forms, see the following chapters:

- Chapter 3, "Basics of Deploying Oracle Forms Applications"
- Chapter 4, "Configuring and Managing Forms Services"
- Chapter 9, "Using Forms Services with Oracle Single Sign-On"
- Chapter 12, "Tracing and Diagnostics"

11.2.3 Securing RADs

To increase the security of RADs and prevent them from being viewable by the OID administrator, perform the following steps:

1. Copy the contents enclosed by `---aci-change.ldif---` into the file `aci-change.ldif`

```
---aci-change.ldif---
dn: cn=Extended Properties,%s_OracleContextDN%
changetype: modify
delete: orclaci
orclaci: access to attr=(orclUserIDAttribute,orclPasswordAttribute) by
guidattr=(orclOwnerGUID)(read,search,compare,write) by
dnattr=(orclresourceviewers) (read,search, compare, write) by
groupattr=(orclresourceviewers) (read,search, write) by * (none)
-
add: orclaci
orclaci: access to attr=(orclUserIDAttribute,orclPasswordAttribute)
DenyGroupOverride by guidattr=(orclOwnerGUID)(read,search,compare,write) by
dnattr=(orclresourceviewers) (read,search, compare, write) by
groupattr=(orclresourceviewers) (read,search, write) by * (none)
---aci-change.ldif---
```

> **Note:** In `aci-change.ldif`, the line beginning with `orclaci: access to attr=` is a single line ending with `by * (none)` and should not have any line breaks in the middle.

2. In the LDIF file, replace `%s_OracleContextDN%` with the distinguished name (DN) of the realm-specific Oracle Context.

   For example, if the DN in the deployment is `dc=acme,dc=com`, then the realm-specific Oracle Context is `cn=OracleContext,dc=acme,dc=com`.

3. Execute the following command on the OID tier:

   ```
   ldapmodify -p <port> -h <host> -D cn=orcladmin -q -v -f
   aci-change.ldif
   ```

4. When this command is run, it will prompt for the `cn=orcladmin` password since the password is not included as a command-line parameter.

To undo these changes, issue the same command (subject to the notes as above), but using the following contents in the `.ldif` file:

```
---aci-revert.ldif---
dn: cn=Extended Properties,%s_OracleContextDN%
```

```
changetype: modify
delete: orclaci
orclaci: access to attr=(orclUserIDAttribute,orclPasswordAttribute)
DenyGroupOverride by guidattr=(orclOwnerGUID)(read,search,compare,write) by
dnattr=(orclresourceviewers) (read,search, compare, write) by
groupattr=(orclresourceviewers) (read,search, write) by * (none)
-
add: orclaci
orclaci: access to attr=(orclUserIDAttribute,orclPasswordAttribute) by
guidattr=(orclOwnerGUID)(read,search,compare,write) by
dnattr=(orclresourceviewers) (read,search, compare, write) by
groupattr=(orclresourceviewers) (read,search, write) by * (none)
---aci-revert.ldif---
```

12

Tracing and Diagnostics

This chapter contains the following sections:

12.1 About Forms Trace

Forms Trace allows you to record information about a precisely defined part of forms functionality or a class of user actions. This is accomplished by defining events for which you want to collect trace information. For example, you can record information about trigger execution, mouse-clicks, or both. From the Enterprise Manager Fusion Middleware Control, you can use trace output to diagnose performance and other problems with Oracle Forms applications.

Forms Trace replaces the functionality that was provided with Forms Runtime Diagnostics (FRD) and Performance Event Collection Services (PECS), which were available in earlier releases of Oracle Forms. Forms Trace allows you to trace the execution path through a form, for example, the steps the user took while using the form.

12.1.1 What Is the Difference between Tracing and Debugging?

You use Forms debugging to find out what happens when a user presses a button. Debugging allows a remote developer to connect to an existing Forms user session and to trace the user actions as the application runs or to debug on a local machine. Forms Trace provides information about the timing of specific events. Oracle Support uses tracing to isolate and analyze issues. For example, you use Forms trace to find out which query takes the longest time to execute, or which trigger causes performance issues with Oracle Forms.

12.2 Enabling and Configuring Forms Trace

An *event* is something that happens inside Oracle Forms as a direct or indirect result of a user action. An example is when a user presses a button that executes a query. An

event set specifies a group of events that you can trace simply by specifying the event set name rather than each event number individually when you start the trace.

Use the **Trace Configuration** selection in the **Forms** menu of Oracle Enterprise Manager page to define the events that you want to trace. This page manages all changes in the `ftrace.cfg` file for you.

Keep these items in mind when working with Forms Trace:

- If you first switch off trace, and then switch it on again with new settings, then trace is enabled with the new trace group.
- In order to trace Forms Processes on Windows, the Process Manager Service needs to have the check box "Allow service to interact with the desktop" selected. When this is not set, attempting to switch on Trace will result in the error: `oracle.sysman.emSDK.emd.comm.RemoteOperationException. Check the User Name and Password.`
- Backup the `ftrace.cfg` and `default.env` files before editing them with Fusion Middleware Control.
- As with most Web applications, it is easy to lose unsaved changes by switching pages. Be sure to save any changes you make through Fusion Middleware Control to Forms configuration, trace, or environment files before proceeding to other pages.

  The length of time it takes for changes to be saved is affected by the number of lines you have changed. For example, an additional fifty lines of comments will take longer to save than just deleting a single entry.

See Section 12.5, "List of Traceable Events" for a list of events and their corresponding event numbers.

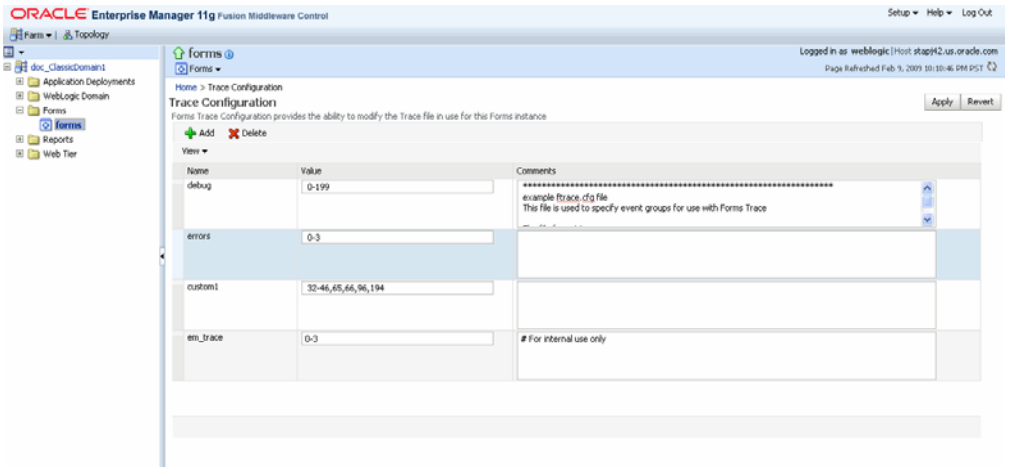
12.2.1 Configuring Forms Trace

To access the Trace Configuration page:

1. Start Fusion Middleware Control.
2. From the Fusion Middleware Control main page, click the link to the Oracle Forms Services instance that you want to configure.
3. From the Forms menu list, select **Trace Configuration**.

   The **Trace Configuration** page (Figure 12–1) is displayed.

Figure 12–1 Trace Configuration Page



To create a new trace group:

1. From the Fusion Middleware Control main page, click the link to the Oracle Forms Services instance that you want to configure.
2. From the Forms menu list, select **Trace Configuration**.

   The **Trace Configuration** page is displayed.
3. Click **Add**.

   The **Add** dialog is displayed.
4. Enter the information for the new trace group:

   Name: Enter a name for the trace group.

   Value: See Table 12–2 for the values of traceable events.

   Comment : Enter a comment.

   - The trace group name must not contain spaces. For example, a_b_c is an acceptable trace group name.
   - There must be a comma between each event number you specify in the Value. For example, 65,66,96,194 is an acceptable value.
   - You can use a range of numbers. For example, 32-46 is an acceptable range.
5. Click **Add**.

   The new trace group is added.
6. Click **Apply** to save the changes, or **Revert** to discard them.

To delete a trace group:

1. In the **Trace Configuration** page, select the group you want to delete.
2. Click **Delete**.

   The trace group is deleted and the **Trace Configuration** page reappears.
3. Click **Apply** to save the changes, or **Revert** to discard them.

To edit an existing trace group:

1. In the **Trace Configuration** page, select the group you want to edit.

2. Enter the value and description for the trace group.
3. Click **Apply** to save the changes, or **Revert** to discard them.

12.2.2 Specifying URL Parameter Options

The following command line parameters are used to configure Forms Trace:

```
Record =
Tracegroup =
Log = <filename>
```

Table 12–1 describes the parameter values:

Table 12–1 Forms Trace Command Line Parameters

Parameter	Values	Description
`Record`	forms	Enables Forms Trace.

`Tracegroup`	Name, event number, or event range	Indicates which events should be recorded and logged.
		■ If Tracegroup is not specified, only error and Startup messages are collected.
		■ Tracegroup is ignored if Forms Trace is not switched on at the command line.
		■ You can create a named set of events using the Tracegroup keyword, for example
		Tracegroup=<keyword>, where <keyword> is specified in ftrace.cfg (for example, Tracegroup=MyEvents).
		This lets you log the events in the named set MyEvents.
		■ You can log all events in a specified range using the Tracegroup keyword, for example
		`Tracegroup = 0-3`
		This lets you log all events in the range defined by 0 <= event <=3.
		■ You can log individual events using the Tracegroup keyword, for example
		`Tracegroup = 34,67`
		■ You can combine event sets using the Tracegroup keyword, for example
		`Tracegroup = 0-3,34,67,SQLInfo`

12.3 Starting and Stopping Forms Trace

You start a trace by specifying trace entries in the URL or from Fusion Middleware Control. Entries should include the grouping of events to collect and the trace file name. Trace collection starts when the form executes.

The following are sample URLs to start a trace:

```
http://example.com/forms/frmservlet?form=cxl&record=forms&tracegroup=0-199
http://example.com/forms/frmservlet?form=cxl&record=forms&tracegroup=mysql
```

To start tracing a session from Fusion Middleware Control:

1. From the Forms menu, select **User Sessions**.

   The **User Sessions** page appears.

2. Select the row containing the Forms user session for which you want to enable tracing.

3. Click **Enable Tracing**.

   The Enable Tracing dialog appears.

4. From the Select Trace Group list, select an available trace group and click **OK**.

   The Enable Tracing dialog is dismissed and tracing is now enabled for the selected Forms user session.

To stop tracing a session from Fusion Middleware Control:

1. From the Forms menu, select **User Sessions**.

   The User Sessions page appears.

2. Select the row containing the Forms user session for which you want to disable tracing.

3. Click **Disable Tracing**.

   The Disable Tracing dialog is displayed.

4. Click **OK**.

   The Disable Tracing dialog is dismissed and tracing is now stopped for the selected Forms user session.

To switch between trace groups for a session:

1. Select the row containing the Forms user session for which you want to change the trace group.

2. Click **Enable Tracing**.

   The Enable Tracing dialog is displayed.

3. From the Select Trace Group list, select the new trace group and click **OK**.

   The Enable Tracing dialog is dismissed. Refresh the page.

12.4 Viewing Forms Trace Output

Only administrators or a user belonging to administrators' group can view trace log files. Once the user has logged in, he or she does not have to log in again in the same browser session to view trace log files for different sessions.

Trace data is stored in a binary file with a `*.trc` extension. The default location of the trace log is `$ORACLE_INSTANCE/FormsComponent/forms/trace/forms_pid.trc` where pid is the process ID of the user session. If you are not using Enterprise Manager Fusion Middleware Control, you need to use the Translate utility. For more information on running the Translate Utility, see the next section Section 12.4.1, "Running the Translate Utility."

To view trace data:

1. From the Forms menu in Fusion Middleware Control, select the **User Sessions** menu item.
2. Select a User Session row and click **Trace Log** to see the contents of the trace log.
3. Log in to view the trace file.

12.4.1 Running the Translate Utility

The Translate utility converts trace data to XML, HTML, or text formats. You need to specify an additional parameter "OutputClass" which has three legal values: "WriteOutTEXT", "WriteOutXML" and "WriteOutHTML". If you do not specify the `outputclass`, the output file is in text format. These values are case-sensitive.

Note: To use the Translate Utility:

1. Set the PATH variable to include the path to the directory containing the Java executable.
2. Set the CLASSPATH variable to include the path to frmxlate.jar.

To convert trace data to Text format:

- At the command line, enter:

```
java oracle.forms.diagnostics.Xlate datafile=a.trc
outputfile=myfile.txt outputclass=WriteOutTEXT
```

This creates a file called myfile.txt in text format.

To convert trace data to HTML format:

- At the command line, enter:

```
java oracle.forms.diagnostics.Xlate datafile=a.trc
outputfile=myfile.html outputclass=WriteOutHTML
```

This creates a file called myfile.html in HTML format.

To convert trace data to XML format:

- To create `myfile.xml`, at the command line, enter:

```
java oracle.forms.diagnostics.Xlate datafile=a.trc
outputfile=myfile.xml outputclass=WriteOutXML
```

This creates a file called myfile.xml in XML format.

12.5 List of Traceable Events

Table 12–2, " List of Traceable Events" lists the events that can be defined for tracing. In future releases of Forms, more events may be added to this list.

Event types are as follows:

- Point event: An event that happens in Oracle Forms as the result of a user action or internal signal for which there is no discernible duration, for example, displaying an error message on the status line. Each instance of this event type creates one entry in the log file.
- Duration event: An event with a start and end, for example, a trigger. Each instance of this event type creates a pair of entries in the log file (a start and end event).

- Built-in event: An event associated with a built-in. Each instance of this event type provides a greater quantity of information about the event (for example, argument values).

Table 12–2 List of Traceable Events

Event Number	Definition	Type
0	Abnormal Error	point
1	Error during open form	point
2	Forms Died Error	point
3	Error messages on the status bar	point
4-31	Reserved for future use	NA
32	Startup	point
33	Menu	point
34	Key	point
35	Click	point
36	Double-click	point
37	Value	point
38	Scroll	point
39	LOV Selection	point
40	not used	not used
41	Window Close	point
42	Window Activate	point
43	Window Deactivate	point
44	Window Resize	point
45	Tab Page	point
46	Timer	point
47	DB Event	point
48	Reserved for future use	NA
49-63	Reserved for future use	NA
64	Form (Start & End)	duration
65	Program Unit (Start & End)	duration
66	Trigger (Start & End)	duration
67	LOV (Start & End)	duration
68	Opening a Editor	point
69	Canvas	point
70	Alert	duration
71	GetFile	point
72-95	Reserved for future use	NA
96	Builtin (Start & End)	builtin

Table 12–2 (Cont.) List of Traceable Events

Event Number	Definition	Type
97	User Exit (Start & End)	duration
98	SQL (Start & End)	duration
99	MenuCreate (Start & End)	duration
100	DB PU (Start & End)	duration
101	Execute Query	duration
102-127	Reserved for future use	N/A
128	Client Connect	point
129	Client Handshake	point
130	Heartbeat	point
131	HTTP Reconnect	point
132	Socket (Start & End)	duration
133	HTTP (Start & End)	duration
134	SSL (Start & End)	duration
135	DB Processing (Start & End)	duration
136	DB Logon (Start & End)	duration
137	DB Logoff (Start & End)	duration
138-159	Reserved for future use	N/A
160-168	Reserved for internal use	N/A
169-191	Reserved for future use	N/A
192*	Environment Dump	N/A
193*	State Delta	N/A
194*	Builtin Arguments	N/A
195*	UserExit Arguments	N/A
196*	Program Unit Arguments.	point
256 and higher	User defined	N/A
1024 and higher	Reserved for internal use	N/A

* These event numbers do not have a `TYPE` because they are not really events, but rather details for events. For example, the State Delta is something you can choose to see - it is triggered by a real action or event.

12.5.1 List of Event Details

The following tables list event details that can be defined for tracing:

- Table 12–3, " User Action Event Details"
- Table 12–4, " Forms Services Event Details"
- Table 12–5, " Detailed Events"

- Table 12–6, " Three-Tier Event Details"
- Table 12–7, " Miscellaneous Event Details"

Note: Event names are case sensitive.

12.5.1.1 User Action Events

Table 12–3 User Action Event Details

Action	Details	Number
Menu Selection	Menu Name, Selection	33
Key	Key Pressed, Form, Block, Item	34
Click	Mouse/Key, Form, Block, Item	35
DoubleClick	Form, Block, Item	36
Value	Form, Block, Item	37
Scroll	Form, Up, Down, Page, Row	38
LOV Selection	LOV Name, Selection Item	39
Alert	AlertName, Selection	40
Tab	Form	45
DB Event	Queue Name	47
Window Activate, Deactivate,Close, Resize	WindowName, FormName, Size	41,42,43,44

12.5.1.2 Forms Services Events

Table 12–4 Forms Services Event Details

Event Name	Details	Number
Form	Form ID, Name, Path, Attached Libraries, Attached Menus	64
Program Unit	Program Unit Name, FormID	65
Trigger	TriggerName, FormName, BlockName, ItemName, FormID	66
LOV	LOV name, FormId	67
Editor	FormId , Editor Name	68
Canvas	FormId , Canvas Name	69

12.5.1.3 Detailed Events

Table 12–5 Detailed Events

Event Name	Details	Number
Builtin	BuiltinName, FormId	96
User Exit	UserExitName, FormId	97
MenuCreate	MenuName, FormID	99
PLSQL	PLSQLSTmt, FormID	100

Table 12–5 (Cont.) Detailed Events

Event Name	Details	Number
ExecQuery	Block Name	101

12.5.1.4 Three-Tier Events

Table 12–6 Three-Tier Event Details

Event Name	Details	Number
Client Connect	Timestamp	128
Client Handshake	Timestamp	129
Heartbeat	Timestamp	130
HTTP Reconnect	NA	131
Socket	FormId, Packets, Bytes	132
HTTP	FormId, Packets, Bytes	133
HTTPS	FormId, Packets, Bytes	134
DB Processing	FormId, Statement	135
DB Logon	FormId	136
DB Logoff	FormId	137

12.5.1.5 Miscellaneous Events

Table 12–7 Miscellaneous Event Details

Event Name	Details	Number
Environment Dump	Selected environment information	192
State Delta	Changes to internal state caused by last action/event	193
Builtin Args	Argument values to a builtin	194
Userexit args	Arguments passed to a userexit	195
Procedure Args	Arguments (in | out) passed to a procedure.	196

12.6 Taking Advantage of Oracle Diagnostics and Logging Tools

Oracle Diagnostics and Logging (ODL) is a feature of Oracle Fusion Middleware that enables administrators to keep a record of all Oracle Forms sessions, monitor Oracle Forms-related network traffic, and debug site configuration problems. Some of the features of Oracle Diagnostics and Logging available to Forms Services include:

- Recording of all Oracle Forms sessions, including session start and end times, and the user's IP address and host name (session-level logging)
- Monitoring of Oracle Forms-related network traffic and performance (session-performance and request-performance-level logging)
- Generating debugging information for site configuration issues (debug-level logging)
- Logging handled through Fusion Middleware Control

- Correlating events in these log files with events in the database
- Automatic handling of log file rotation.
- Handling of log size restriction by the mechanism rather than by OS level scripts as was done previously

These sections on the servlet logging tools contain the following:

- Section 12.6.1, "Enabling Oracle Diagnostics and Logging"
- Section 12.6.4, "Location of Log Files"
- Section 12.6.5, "Example Output for Each Level of Servlet Logging"

12.6.1 Enabling Oracle Diagnostics and Logging

When you turn on logging, the Listener Servlet writes log messages to the servlet log file. Examples of output for the various levels of logging are in Section 12.6.5, "Example Output for Each Level of Servlet Logging".

Table 12–8 describes the supported logging capabilities. If no string is appended to serverURL, then default logging is supported. To start other loggers, they must be specified in serverURL as described in the next section.

Table 12–8 Supported logging capabilities

String appended to serverURL client parameter	Description of logging
(none)	During Forms servlet initialization, a message is written to the log file stating the name and path of the configuration file being used. Messages of levels higher and equal to the log level set for the default logger in logging.xml are logged. Default Value is set to NOTIFICATION:1 and levels NOTIFICATION:1, WARNING:1, ERROR:1 and INTERNAL_ERROR are logged.
/session	Log messages are written whenever a Forms session starts or ends. These give the host name and IP address of the client (the computer on which the user's Web browser is running), the runtime process id, and a unique internal session id number.
/sessionperf	Performance summary statistics are included with the session end message.
/perf	A performance message is written for every request from the client.
/debug	Full debug messages. Other debug messages are written in addition to the messages mentioned above. This logging level is verbose and is intended mainly for debugging and support purposes.

12.6.1.1 Specifying Logging

To specify logging for all users, change the serverURL entry in the default section in the **Web Configuration** page to the following:

```
serverURL=/forms/lservlet/<string>
```

where <string> specifies the logging capability as defined in Table 12–8. If no string is provided, the default logging For example, if you want to start session-level logging, modify the serverURL as follows:

```
serverURL=/forms/lservlet/session
```

12.6.1.2 Specifying Logging Levels Using Fusion Middleware Control

To set the log levels for Forms servlet logging using Fusion Middleware Control, perform the following:

1. From the Fusion Middleware Control, select the managed server (for example WLS_FORMS).
2. From the WebLogic Server menu, select Logs, then Log Configuration.
3. In the Logger Name field, expand Root Logger. Expand each of the following: oracle, oracle.forms. The Logger name defined in serverURL as described in Section 12.6.1.1, "Specifying Logging" is displayed, for example (oracle.forms.servlet.debug).
4. Choose the Log level as required from the list in the Oracle Diagnostic Logging Level field. Refer to Table 12–9 for the mapping of the internal Forms log level to the Java levels.

Table 12–9 Oracle Diagnostic Logging Levels

Internal Forms Log Levels	Java Log Levels
DEBUG	TRACE:32
REQUEST_PERFORMANCE	TRACE:16
SESSION_PERFORMANCE	TRACE:1
SESSION_START_END	NOTIFICATION:16
NOTIFICATION	NOTIFICATION:1
WARNING	WARNING:1
ERROR	ERROR:1
INTERNAL_ERROR	INTERNAL_ERROR

This configuration modifies the logging.xml file for the managed server.

12.6.1.3 Specifying Full Diagnostics in the URL that Invokes the Forms Servlet

To start full diagnostics, specify the parameter serverURL in `formsweb.cfg` as follows:

```
serverURL=/forms/lservlet/debug
```

Start the Oracle Forms application using a URL as follows:

```
http://example.com/forms/frmservlet/debug?
```

12.6.2 Viewing Diagnostics Logs

You view the contents of diagnostics logs from Fusion Middleware Control.

To view the contents of diagnostics logs:

1. From the Forms menu, select Home.

   The Fusion Middleware Control home page is displayed.
2. In the Forms Deployment region, scroll to the Servlet Logs column.
3. Click the corresponding Logs link for the target deployed application.

   The Log Messages page is displayed.

12.6.3 Using the Servlet Page

From the Forms menu, select Monitoring and then Servlet Logs. Use this page to search, sort, view, download, and export collected server diagnostics logs.

For additional information on managing and viewing the log files, refer to the *Oracle Fusion Middleware Administrator's Guide*.

12.6.4 Location of Log Files

The default servlet log file is named `formsapp-diagnostic.log`. It is written to the `WLS_FORMS/logs` directory of the Oracle WebLogic Managed Server to which Forms is deployed.

In Oracle Forms Services, the full path is:

```
$DOMAIN_HOME/servers/WLS_FORMS/logs/<application
name>-diagnostic.log
```

The trace logs are stored in files named `forms_pid.trc` by default, where `pid` is the process ID of the user session. The default location of the trace log is:

```
$ORACLE_INSTANCE/FormsComponent/forms/trace/forms_pid.trc
```

Use the Translate Utility described in Section 12.4.1, "Running the Translate Utility" to view them.

12.6.5 Example Output for Each Level of Servlet Logging

The following are examples of the type of output you get when you use the following levels of logging:

- (none)
- /session
- /sessionperf
- /perf
- /debug

12.6.5.1 (none)

```
[2008-09-10T06:58:47.106-07:00] [WLS_FORMS] [NOTIFICATION] [FRM-93100]
 [oracle.forms.servlet] [tid: 11] [ecid: 0000HlCYKnmD4i8nvgy0V1181x4u000000,0]
 [APP: formsapp] [arg:
configFileName:     <configfilename>
testMode:           false] Initializing the Forms Servlet.  Initialization
 parameters are:[[
    configFileName:    <configfilename>
    testMode:
false
]]
[2008-09-10T06:58:53.517-07:00] [WLS_FORMS] [NOTIFICATION] [FRM-93180]
 [oracle.forms.servlet] [tid: 11] [ecid: 0000HlCZfTDD4i8nvgy0V1181x4u000003,0]
 [APP: formsapp] [arg:
envFile:            null
WorkingDirectory:   null
executable:         null
WaitTime:           500
MaxBlockTime:       1000]
Initializing ListenerServlet.  Initialization parameters
```

```
 are:[[
    envFile:            null
    WorkingDirectory:   null
    executable:         null
    WaitTime:           500
    MaxBlockTime:       1000
]]
```

12.6.5.2 /session

```
[2008-09-11T07:35:01.507-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93251]
[oracle.forms.servlet.session] [tid: 14] [ecid:
0000HlHpYGDD4i8nvgy0V118mFuv00000V,0] [SRC_CLASS:
oracle.forms.servlet.RunformSession] [APP: formsapp] [SRC_METHOD: <init>] [FORMS
SESSION_ID: ..8] [arg: supadhya-pc1] [arg: 10.177.254.46] Runtime session started
for client <pc1> (IP address <ip address>).
2008-09-11T07:35:01.798-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93548]
[oracle.forms.servlet.session] [tid: 14] [ecid:
0000HlHpYGDD4i8nvgy0V118mFuv00000V,0] [SRC_CLASS:
oracle.forms.servlet.RunformProcess] [APP: formsapp] [SRC_METHOD: connect] [FORMS
SESSION_ID: ..8] [arg: 7765] Runtime process ID is 7765.
2008-09-11T07:38:11.372-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93252]
[oracle.forms.servlet.session] [tid: 14] [ecid:
0000HlHpYGDD4i8nvgy0V118mFuv00000V,0] [SRC_CLASS:
oracle.forms.servlet.RunformSession] [APP: formsapp] [SRC_METHOD: stop] [FORMS
SESSION_ID: ..8] Forms session ended.
```

12.6.5.3 /sessionperf

```
[2008-09-11T07:40:25.923-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93251]
[oracle.forms.servlet.sessionperf] [tid: 17] [ecid:
0000HlHqlS9D4i8nvgy0V118mFuv00000Y,0] [SRC_CLASS:
oracle.forms.servlet.RunformSession] [APP: formsapp] [SRC_METHOD: <init>] [FORMS
SESSION_ID: ..9] [arg: <pc1>] [arg: 10.177.254.46] Runtime session started
for client <pc1> (IP address 10.177.254.46).
2008-09-11T07:40:26.223-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93548]
 [oracle.forms.servlet.sessionperf] [tid: 17] [ecid:
0000HlHqlS9D4i8nvgy0V118mFuv00000Y,0] [SRC_CLASS:
oracle.forms.servlet.RunformProcess] [APP: formsapp] [SRC_METHOD: connect] [FORMS
SESSION_ID: ..9] [arg: 8023] Runtime process ID is 8023.
2008-09-11T07:40:43.593-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93252]
[oracle.forms.servlet.sessionperf] [tid: 17] [ecid:
0000HlHqlS9D4i8nvgy0V118mFuv00000Y,0] [SRC_CLASS:
oracle.forms.servlet.RunformSession] [APP: formsapp] [SRC_METHOD: stop] [FORMS
SESSION_ID: ..9] Forms session ended.
[2008-09-11T07:40:43.594-07:00] [WLS_FORMS] [TRACE] [FRM-93710]
[oracle.forms.servlet.sessionperf] [tid: 17] [ecid:
0000HlHqlS9D4i8nvgy0V118mFuv00000Y,0] [SRC_CLASS:
oracle.forms.servlet.RunformSession] [APP: formsapp] [SRC_METHOD: stop] [FORMS
SESSION_ID: ..9] [arg: 1.557] [arg: 6] [arg: 0] [arg: 1.000] [arg: 0.259] [arg:
5106] [arg: 352] Total duration of network exchanges is 1.557.[[
Total number of network exchanges is 6 (0 long ones over 1.000 sec).
Average time for one network exchange (excluding long ones) is 0.259.
Total number of bytes sent is 5106.
Total number of bytes received is 352.
]]
```

12.6.5.4 /perf

```
[2008-09-11T07:42:46.560-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93251]
```

```
[oracle.forms.servlet.perf] [tid: 14] [ecid: 0000HlHrJmWD4i8nvgy0V118mFuv00000^,0]
[SRC_CLASS: oracle.forms.servlet.RunformSession] [APP: formsapp] [SRC_METHOD:
<init>] [FORMS_SESSION_ID: ..10] [arg: <pc1>] [arg: 10.177.254.46] Runtime
session started for client <pc1> (IP address <ip address>).
[2008-09-11T07:42:46.854-07:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93548]
[oracle.forms.servlet.perf] [tid: 17] [ecid: 0000HlHqlS9D4i8nvgy0V118mFuv00000Y,0]
[SRC_CLASS: oracle.forms.servlet.RunformProcess] [APP: formsapp] [SRC_METHOD:
connect] [FORMS_SESSION_ID: ..10] [arg: 8149] Runtime process ID is 8149.
[2008-09-11T07:42:46.865-07:00] [WLS_FORMS] [TRACE:16] [FRM-93700]
[oracle.forms.servlet.perf] [tid: 17] [ecid: 0000HlHqlS9D4i8nvgy0V118mFuv00000Y,0]
[SRC_CLASS: oracle.forms.servlet.ListenerServlet] [APP: formsapp] [SRC_METHOD:
doPost] [FORMS_SESSION_ID: ..10] [arg: 0.011] [arg: 8] [arg: 8] [arg: null]
Request duration is 0.011 seconds.  Request size is 8 bytes; response size is 8
bytes.
[2008-09-11T07:42:47.921-07:00] [WLS_FORMS] [TRACE:16] [FRM-93700]
[oracle.forms.servlet.perf] [tid: 17] [ecid: 0000HlHqlS9D4i8nvgy0V118mFuv00000Y,0]
[SRC_CLASS: oracle.forms.servlet.ListenerServlet] [APP: formsapp] [SRC_METHOD:
doPost] [FORMS_SESSION_ID: ..10] [arg: 0.438] [arg: 272] [arg: 5022] [arg: null]
Request duration is 0.438 seconds.  Request size is 272 bytes; response size is
5022 bytes.
```

12.6.5.5 /debug

```
[2009-02-11T14:39:03.016+00:00] [WLS_FORMS] [NOTIFICATION:16] [FRM-93250]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_lhDcD4i8nvgy0V119Xz350000HZ,0] [APP: formsapp#11.1.1] Forms session started.
[2009-02-11T14:39:03.017+00:00] [WLS_FORMS] [TRACE:32] [FRM-94200]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_lhDcD4i8nvgy0V119Xz350000HZ,0] [SRC_CLASS: oracle.forms.servlet.FormsServlet]
 [APP: formsapp#11.1.1] [SRC_METHOD: doRequest] [FORMS_SESSION_ID: ..43] [arg:
 GET] [arg:
cmd:                frmservlet
config:             null
requestCharset:     null
QueryString:        null
Content-Type:       null
Accept-Charset:     null
responseCharset:    null] FormsServlet receiving GET request.  Details:[[
    cmd:                frmservlet
    config:             null
    requestCharset:     null
    QueryString:        null
    Content-Type:       null
    Accept-Charset:     null
    responseCharset:    null
]]
[2009-02-11T14:39:03.017+00:00] [WLS_FORMS] [TRACE:32] [FRM-94281]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_lhDcD4i8nvgy0V119Xz350000HZ,0] [SRC_CLASS: oracle.forms.servlet.ListenerServlet]
 [APP: formsapp#11.1.1] [SRC_METHOD: printSessionDetails] [FORMS_SESSION_ID: ..43]
 No current servlet session ID.
[2009-02-11T14:39:03.017+00:00] [WLS_FORMS] [TRACE:32] [FRM-94170]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_lhDcD4i8nvgy0V119Xz350000HZ,0] [SRC_CLASS: oracle.forms.servlet.FormsServlet]
 [APP: formsapp#11.1.1] [SRC_METHOD: findFile] [FORMS_SESSION_ID: ..43] [arg:
 basejpi.htm] [arg: <config folder>] File basejpi.htm is missing from the
```

```
 current directory, looking in <config folder>
[2009-02-11T14:39:21.460+00:00] [WLS_FORMS] [TRACE:32] [FRM-94200]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llhoD4i8nvgy0V119Xz350000Hd,0] [SRC_CLASS: oracle.forms.servlet.FormsServlet]
 [APP: formsapp#11.1.1] [SRC_METHOD: doRequest] [FORMS_SESSION_ID: ..43] [arg:
 GET] [arg:
cmd:                startsession
config:             null
requestCharset:     null
QueryString:
ifsessid=..43&acceptLanguage=en-us&ifcmd=startsession&iflocale=en-US
    Content-Type:       null
Accept-Charset:     null
responseCharset:    null]
FormsServlet receiving GET request.  Details:[[
    cmd:                startsession
    config:             null
    requestCharset:     null
    QueryString:
ifsessid=..43&acceptLanguage=en-us&ifcmd=startsession&iflocale=en-US
    Content-Type:       null
    Accept-Charset:     null
    responseCharset:    null
]]


.
.
.
.

[2009-02-11T14:39:21.716+00:00] [WLS_FORMS] [TRACE:32] [FRM-94201]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llloD4i8nvgy0V119Xz350000Hf,0] [SRC_CLASS: oracle.forms.servlet.ListenerServlet]
[APP: formsapp#11.1.1] [SRC_METHOD: doGet] [FORMS_SESSION_ID: ..43] [arg: GET]
 [arg:
cmd:                getinfo
QueryString:        ifcmd=getinfo&ifhost=supadhya-pc1&ifip=10.177.254.239]
 ListenerServlet receiving GET request.  Details:[[
    cmd:                getinfo
    QueryString:        ifcmd=getinfo&ifhost=supadhya-pc1&ifip=10.177.254.239
]]

[2009-02-11T14:39:21.717+00:00] [WLS_FORMS] [TRACE:32] [FRM-94282]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llloD4i8nvgy0V119Xz350000Hf,0] [SRC_CLASS: oracle.forms.servlet.ListenerServlet]
 [APP: formsapp#11.1.1] [SRC_METHOD: printSessionDetails] [FORMS_SESSION_ID: ..43]
 [arg:
 HyLhJSjZ85F5GWbZLDgwp1MY02FK5tC6yVDP1LylbCvgmv9y3CfK!126690176!1234363161461]
 Existing servlet session, ID =
 HyLhJSjZ85F5GWbZLDgwp1MY02FK5tC6yVDP1LylbCvgmv9y3CfK!126690176!1234363161461
[2009-02-11T14:39:21.717+00:00] [WLS_FORMS] [TRACE:32] [FRM-94286]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llloD4i8nvgy0V119Xz350000Hf,0] [SRC_CLASS: oracle.forms.servlet.ListenerServlet]
 [APP: formsapp#11.1.1] [SRC_METHOD: printSessionDetails] [FORMS_SESSION_ID: ..43]
 Session ID is not from cookie.
```

```
[2009-02-11T14:39:21.717+00:00] [WLS_FORMS] [TRACE:32] [FRM-94430]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llloD4i8nvgy0V119Xz350000Hf,0] [SRC_CLASS: oracle.forms.servlet.RunformSession]
 [APP: formsapp#11.1.1] [SRC_METHOD: <init>] [FORMS_SESSION_ID: ..43] Trying to
 get a prestarted process.
[2009-02-11T14:39:21.717+00:00] [WLS_FORMS] [TRACE:32] [FRM-94432]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llloD4i8nvgy0V119Xz350000Hf,0] [SRC_CLASS: oracle.forms.servlet.RunformSession]
 [APP: formsapp#11.1.1] [SRC_METHOD: <init>] [FORMS_SESSION_ID: ..43] Prestarted
 process is not available.
[2009-02-11T14:39:21.718+00:00] [WLS_FORMS] [TRACE:32] [FRM-94522]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llloD4i8nvgy0V119Xz350000Hf,0] [SRC_CLASS: oracle.forms.servlet.RunformSession]
 [APP: formsapp#11.1.1] [SRC_METHOD: <init>] [FORMS_SESSION_ID: ..43] [arg: null]
 Creating new runtime process using default executable.
[2009-02-11T14:39:21.718+00:00] [WLS_FORMS] [TRACE:32] [FRM-94532]
 [oracle.forms.servlet] [tid: [ACTIVE].ExecuteThread: '2' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>] [ecid: 0000Hx
_llloD4i8nvgy0V119Xz350000Hf,0] [SRC_CLASS: oracle.forms.servlet.RunformProcess]
 [APP: formsapp#11.1.1] [SRC_METHOD: startProcess] [FORMS_SESSION_ID: ..43] [arg:
 frmweb webfile=HTTP-0,default] RunformProcess.startProcess():  executing frmweb
 webfile=HTTP-0,default

.
.
.
.
```

13

Upgrading to Oracle Forms Services 11*g*

This chapter describes the upgrade process from Forms 6*i*. For information about changed or obsolete features, see the *Oracle Forms Upgrading Oracle Forms 6i to Oracle Forms 11g Guide*.

This chapter contains the following sections:

- Section 13.1, "Oracle Forms Services Upgrade Items"
- Section 13.2, "Oracle Forms Services Upgrade Tasks"
- Section 13.3, "Validating the Oracle Forms Services Upgrade"

For upgrading from Oracle Forms 10*g* and prior releases, you can use the Upgrade Assistant. Refer to the following documents for more information.

- *Oracle Fusion Middleware Upgrade Planning Guide*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Portal, Forms, Reports, and Discoverer*

13.1 Oracle Forms Services Upgrade Items

Table 13–1 describes the items that are upgraded. These items include files, executables, or settings that you must add, change, delete, or replace in the Oracle Forms Services installation.

Table 13–1 Oracle Forms Services Upgrade Items

Upgrade Item	Location in 6i Oracle home	Location in 11*g* (11.1.1) Oracle home	Description and Notes
Oracle HTTP Server configuration file: `6iserver.conf` (upgrades to `forms.conf`)	`6iserver/conf/`	`$ORACLE_INSTANCE/config/OHS/<OHS Instance>/moduleconf/forms.conf`	Contains virtual path mappings.
Servlet environment file: `default.env`	`6iserver/forms60/server`	`$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/default.env`	Contains environment variables settings for the Forms servlet Runtime Process.
Configuration files with Forms servlet alias: `jserv.properties` (upgrades to `web.xml`)	`/Apache/jserv/conf`	`$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF`	Contains Forms servlet aliases.
Application configuration file: `formsweb.cfg`	`6iserver/forms60/server`	`$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/formsweb.cfg`	Contains Forms Services application configuration information.
Forms servlet template html files: (`*.htm`, `*.html`)	`6iserver/forms60/server`	`$ORACLE_INSTANCE/config/FormsComponent/forms/server/`	Default and user defined Forms servlet template HTML files.
Forms application modules (`fmb`/`fmx` files)			Forms modules (`fmb` and `fmx` files) deployed to Oracle 6i Forms Services must be upgraded to be deployed to Oracle Forms Services. Note that when you upgrade to 11*g*, you need to update the `FORMS_PATH` variable with the location of the `fmx` file. For more information on `FORMS_PATH`, see Section 4.3, "Managing Environment Variables".

13.2 Oracle Forms Services Upgrade Tasks

This section explains how to perform the Oracle Forms Services upgrade. It is divided into the following sub-sections:

- Section 13.2.1, "Upgrade Recommendations and Troubleshooting Tips" on page 13-3
- Section 13.2.2, "Upgrading Oracle Forms Services Application Modules" on page 13-3
- Section 13.2.3, "Upgrading Common Gateway Interface (CGI) to the Oracle Forms Servlet" on page 13-4
- Section 13.2.4, "Upgrading Static HTML Start Files to Generic Application HTML Start Files" on page 13-5
- Section 13.2.5, "Upgrading the Forms 6i Listener to the Forms Listener Servlet" on page 13-7

13.2.1 Upgrade Recommendations and Troubleshooting Tips

Consider the following recommendations and considerations while upgrading Forms applications:

- Keep the Oracle6*i* Forms Services installation available until applications are successfully deployed and tested.
- Back up and secure all files, then upgrade the source files.
- Replace `Run_Product` calls to integrated Reports with `Run_Report_Object` calls to Oracle Reports (or use the PL/SQL conversion utility, Forms Migration Assistant in Oracle Forms).
- Install Oracle Fusion Middleware and configure the `formsweb.cfg` file with the information used by your applications.
- Copy the environment files used by the applications to the same relative directory.
- Copy the upgraded Oracle Forms application module files to the computer on which Oracle WebLogic Server is installed, if it is not the same computer.
- After starting Oracle WebLogic Server, access the Forms Services Listener Servlet test page with this URL (default port 8888):

  `http://<hostname>:<port>/forms/frmservlet?form=test.fmx`
- Verify that any application settings are added to the `formsweb.cfg` file and that the environment variable `Forms_Path` contains the directory of the application modules.
- Verify that you can connect to the database using SQL*Plus.
- Use the following URL to invoke upgraded applications:

  `http://<hostname>:<port>/forms/frmservlet?config=<your application name>`

13.2.2 Upgrading Oracle Forms Services Application Modules

This section provides instructions for upgrading from Forms Application Modules (`fmb` files) that were deployed in Oracle 6*i* Forms Services. Follow these steps to upgrade Forms Application Modules (`fmb` files) deployed in Oracle 6*i* Forms Services to an Oracle Forms Services installation.

1. Copy the Forms application files to a new directory.
2. Optionally, use the Forms Migration Assistant to upgrade the Forms Application Modules (`.fmb` files), Forms menu modules (`.mmb` files), and the Library modules (`.pll` files).
3. Use the Forms Compiler (`frmcmp.sh` on Unix or `frmcmp.exe` on Windows) to regenerate the Forms Application executable files (`fmx`, `mmx`, and `plx` files).

For more information, see Oracle Forms Upgrading Oracle Forms 6*i* to Oracle Forms 11*g* at:

`http://www.oracle.com/technology/documentation/`

13.2.3 Upgrading Common Gateway Interface (CGI) to the Oracle Forms Servlet

This section provides instructions to upgrade Forms CGI to the Forms servlet deployment. Follow these steps if you are using the Oracle 6i Forms Services Common Gateway Interface to dynamically render the Forms Applet start HTML file for applications.

CGI deployment for Forms applications was introduced in Oracle Forms Services Release 6*i* to enable the Forms Applet Start HTML file to render dynamically. Forms CGI uses the `formsweb.cfg` configuration file and an HTML template to create the start HTML file for an application. The CGI interface is configured by an entry in the Forms HTTP configuration file `6iserver.conf` (it is referenced by an `Include` directive in the Oracle HTTP Server `oracle_apache.conf` file), which contains a `ScriptAlias` directive identifying `dev60cgi` for the directory structure containing the `ifcgi60.exe` file.

The Forms servlet renders the HTML in the same manner as the CGI, but also provides an automatic browser type detection. The Forms servlet is configured when you install Oracle Forms Services, and is named `frmservlet`.

To access the Forms servlet, request the URL:

`http://<`*`hostname`*`>:<`*`port`*`>/forms/frmservlet`

This URL is similar to the URL used with the CGI Interface in Oracle 6*i* Forms Services. To call an application configured as `myapp` in the custom configuration section of the `formsweb.cfg` file, request the URL:

`http://<`*`hostname`*`>:<`*`port`*`>/forms/frmservlet?config=myapp`

The Forms servlet is automatically configured during installation. The installer creates a virtual path `/forms/` pointing to the Oracle Forms Services configuration, `formsapp` and `formsweb`.

Follow these steps to upgrade an Oracle 6*i* Forms Services Release 6*i* CGI environment to an Oracle Forms Services servlet environment:

1. Copy all of the application-specific configurations from `<`*`source_OH`*`>/Forms60/Server/formsweb.cfg` and append them to `<`*`destination_Domain_Dir`*`>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/formsweb.cfg`.

   Note: Do not copy and replace the entire `formsweb.cfg` file in `<`*`source_OH`*`>` to `<`*`destination_Domain_Dir`*`>`. The file in Release 6*i* is different from the Oracle Forms Services file. Copy only the application configuration to `<`*`destination_Domain_Dir`*`>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/formsweb.cfg`.

2. Configure `Forms_Path` in the `default.env` file to point to the upgraded Oracle Forms Services application modules.

   Note: You can create a new environment file by copying `default.env`, modifying it for use with a particular application, and adding `envFile=<`*`created environment file`*`>` to the custom application section in the `formsweb.cfg` file.

3. If you changed the Oracle 6*i* Forms HTML template files, then make the same changes to the Oracle Forms Services HTML template files.

> **Note:** You must make these changes in `basejpi.htm` rather than `basejini.htm` because the servlet supports the Sun Java plug-in.

13.2.4 Upgrading Static HTML Start Files to Generic Application HTML Start Files

Each application deployed to Oracle Forms Services has a custom application definition, configured in the `formsweb.cfg` configuration file. It automatically inherits the general system settings, such as the names and locations of the base HTML template files.

The name of the custom application definition becomes part of the Forms application URL. The following custom settings define two different applications:

```
[MyHR_app]
serverURL=/forms/lservlet
Form = hr_main.fmx
lookAndFeel=oracle
Otherparams=myParam1=12
Userid=scott/tiger@orcl
```

The following URL invokes this application:

`http://<hostname>:<port>/forms/frmservlet?config=MyHR_app`

Another custom application definition might look like this:

```
[booking_app]
ServerURL=/forms/lservlet
Form = book.fmx
lookAndFeel=oracle
Otherparams=
Userid=
```

The following URL invokes this application:

`http://<hostname>:<port>/forms/frmservlet?config=booking_app`

For each static HTML file, you must create a custom application definition. Part of the static HTML file is the `archive` parameter directive, specifying at least the `frmall.jar` file in Oracle Forms Services. If you added a custom archive file, then the `archive` parameter directive would resemble the following: `Archive=frmall.jar,custom.jar`. Using the Forms servlet and the `formsweb.cfg` file, the archive settings are defined under the User Parameter section. All custom application settings inherit these values, so you don't have to explicitly set this parameter, unless you add a `custom.jar` file as required by an application.

If `custom.jar` was added, then you can add the following lines to the custom application definition. The example below assumes that you are using another VM.

```
[booking_app]
archive=frmall.jar, custom.jar
ServerURL=/forms/lservlet
Form = book.fmx
lookAndFeel=oracle
Otherparams=
Userid=
```

Follow these steps to upgrade applications:

1. Edit the `default.env` file, adding the location of the Oracle Forms Services application modules to the `Forms_Path`.
2. Edit the `formsweb.cfg` file, appending a custom application section for each static HTML application that you want to replace.
3. Name each custom application section, using a name that contains no spaces and is enclosed in square brackets, for example: `[booking_app]`, `[MyHR_app]`.
4. Start the application using this URL:

   `http://<hostname>:<port>/forms/frmservlet?config=<name>`

13.2.4.1 Using Static HTML Files with Oracle Forms Services

If you need to, you can continue to use static HTML files in Oracle Forms Services. However, with static HTML files, some features (Single Sign-On) are not available for use by Forms applications.

The Forms Listener servlet by default points to `/forms/lservlet` after installation. To use static HTML files in Oracle Forms Services, you must modify each static start HTML file to include a value for the `serverURL` parameter. The `serverPort` and `serverHost` parameters are no longer used, and can be left undefined.

Follow these steps to use static HTML files with Oracle Forms Services:

1. Configure `Forms_Path` in the `default.env` file to point to the upgraded Oracle Forms Services application modules.
2. Create virtual directories in the `$ORACLE_INSTANCE/config/OHS/<OHS Instance>/moduleconf/forms.conf` file to point to the location of the static HTML start files.
3. Modify the application start HTML files as follows:
   a. Add the `serverURL` value `/forms/lservlet`.
4. Change the `codebase` parameter to `forms/java`.
5. Navigate to `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF` and edit the `web.xml` file.
6. Set the envFile initialization parameter for the Listener Servlet to point to the environment file (usually `default.env`).

After editing, the entry in the `web.xml` file for the Forms listener servlet should resemble the following:

```
<!--Forms listener servlet-->
<servlet>
  <servlet-name>lservlet</servlet-name>
     <servlet-class>oracle.forms.servlet.ListenerServlet</servlet-class>
  <init-param>
     <param-name>envFile</param-name>
     <param-value>destination_Domain_Dir/forms/server/default.env</param-value>
  </init-param>
</servlet>
```

13.2.5 Upgrading the Forms 6*i* Listener to the Forms Listener Servlet

The Forms 6*i* Listener was a C program that starts a Forms runtime process on behalf of an incoming Forms Web request. The Forms Web runtime process was then directly accessed by the Forms client applet, using a direct socket or an HTTP socket connection. The Forms Listener was then no longer involved in the application Web client-server communication process, and was free to handle other incoming Web requests.

The Forms Listener servlet, a Java program, also takes incoming Web requests for a Forms application and starts the Forms Web runtime process. Unlike the Forms 6*i* Listener, the Forms Listener servlet remains between the Forms application applet-server communication.

While the Forms 6*i* Listener listened on a specific port (by default, 9000), the Forms servlet does not need an extra port, and is accessed by the HTTP listener port. The Forms Listener servlet was introduced in the Forms 6*i* patch 4, and is the only listener supported in Forms Services.

The Forms Listener servlet is automatically configured during the installation. The installer creates a virtual path `/forms/` pointing to the Oracle Forms Services configuration, `formsapp` and `formsweb`.

To access the Forms Listener servlet test form, request the following URL:

`http://<hostname>:<port>/forms/frmservlet?form=test.fmx`

Ability to access this page means that the Forms Listener servlet is configured and ready to use. `frmservlet` is the access name configured for the Forms servlet during installation. The name of the Listener Servlet is `lservlet`.

If the Forms Listener servlet is accessed with the Forms servlet, then only the custom application settings from the `Forms60/server/formsweb.cfg` file need to be appended to the `formsweb.cfg` file. All application configurations automatically inherit the `serverURL` parameter value `/forms/lservlet` from the global system parameter settings.

To change a Forms application deployment from the Forms Listener architecture to the Listener Servlet architecture, you need only supply a value for the `serverURL` parameter in the `formsweb.cfg` file. During installation, this parameter is set to `/forms/lservlet`.

Follow these steps to upgrade to the Forms Listener servlet:

1. Copy the Forms application files to a new directory and upgrade them to Oracle Forms Services modules as described in Section 13.2.2, "Upgrading Oracle Forms Services Application Modules". on page 13-3.
2. Edit the `forms/server/default.env` file to add the location of the upgraded Forms application modules to the `Forms_Path` variable.
3. Copy all of the custom application settings from `<source_OH>/Forms60/Server/formsweb.cfg` and append them to `<destination_Domain_Dir>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/formsweb.cfg`.
4. If an application requires its own environment file, then instead of defining a separate servlet alias for the Listener Servlet, set the `envFile` parameter in the custom application definition section in `<destination_Domain_Dir>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/formsweb.cfg` to point to the new environment file. For example:

```
envFile=myEnvFile.env
```

where `myEnvFile.env` is located in the `forms/server` directory.

5. If you changed the Oracle 6*i* Forms Services HTML template files, then make the same changes to the Oracle Forms Services HTML template files.

   Note: If you need to change the underlying HTML files, you should make a copy of the provided template files before editing them. Save the edited HTML files under a different name, and leave the default templates provided with the installation unchanged. This prevents overwriting of your customized HTML template files when patch sets are applied to the application.

   To use your own template files with applications, use these parameters in the system section, or one of your custom application definitions: `baseHTML=<your base template>.htm`

6. Start the application with this URL:

   `http://<hostname>:<port>/forms/frmservlet?config=<application>`

13.2.6 Upgrading the Forms Listener Servlet Architecture to Oracle Forms Services

In Oracle9*i*AS Forms Services Release 6*i*, the Listener Servlet, if not aliased, is accessed by the `oracle.forms.servlet.ListenerServlet`. The Listener Servlet configuration exists in the `jserv.properties` file and the `zone.properties` file.

In Oracle Forms Services, the Forms Listener servlet is the same except for the servlet names, which are `frmservlet` and `lservlet`, and the servlet container. The configuration is performed during installation. The Listener Servlet configuration in Oracle WebLogic Managed Server is stored in `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF/web.xml`. Some initialization parameters, like the `envFile` parameter, need no longer be configured with the servlet engine, because they are moved to the `formsweb.cfg` file.

The Forms Listener servlet is automatically configured during the Oracle WebLogic Server installation. The installer creates a virtual path `/forms/` pointing to the Oracle Forms Services configuration, `formsapp` and `formsweb`.

To access the Forms Listener servlet test form, request the following URL:

`http://<hostname>:<port>/forms/frmservlet?form=test.fmx`

Ability to access this page means that the Forms Listener servlet is configured and ready to use. `frmservlet` is the access name configured for the Forms servlet during installation. The name of the Listener Servlet is `lservlet`.

Follow these steps to upgrade the Listener Servlet architecture to Oracle Forms Services:

1. Copy the Forms application files to a new directory and upgrade them to Oracle Forms Services modules.
2. Edit the `forms/server/default.env` file, adding the location of the upgraded Forms application modules to the `Forms_Path` variable.

3. Copy all of the custom application settings from `<source_OH>/Forms60/Server/formsweb.cfg` and append them to `<destination_Domain_Dir>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/formsweb.cfg`.
4. If an application requires its own environment file, then instead of defining a servlet alias for the Listener Servlet, set the envFile parameter in the custom application definition section in `<destination_Domain_Dir>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/formsweb.cfg` to point to the new environment file. For example:

   ```
   envFile=myEnvFile.env
   ```

   where `myEnvFile.env` is located in the `forms/server` directory.
5. If you changed the Forms Services Release 6*i* HTML template files, then make the same changes to the Oracle Forms Services HTML template files.

   Note: If you need to change the underlying HTML files, you should make a copy of the provided template files before editing them. Save the edited HTML files under a different name, and leave the default templates provided with the installation unchanged. This prevents overwriting of your customized HTML template files when patch sets are applied to the application.

   To use your own template files with applications, use these parameters in the system section, or one of your custom application definitions: baseHTML=<your base template>.htm
6. Start the application with this URL:

   ```
   http://<hostname>:<port>/forms/frmservlet?config=<application>
   ```

13.2.7 Upgrading Load Balancing

The method of upgrading the load balancing in Forms Services 6*i* depends on the deployment method used.

- With the Forms 6*i* listener, the Metrics Server (a separate process) performs load balancing.
- With the Forms 6*i* servlet, load balancing is configured with the JServ servlet engine, using round robin load balancing among JServ engines.
- In Oracle Forms Services, load balancing is managed by Oracle WebLogic Managed Server process. It binds Web requests to the servlet container processing the Forms servlet and the Forms Listener servlet.

13.2.8 Usage Notes

This section contains hints and tips that may be useful in the upgrade.

13.2.8.1 Deploying Icon Images with the Forms Servlet

Using static HTML start files in Forms Services Release 6*i* allowed storage of images in a location relative to the start HTML file. The Forms servlet in Oracle Forms Services does not support this.

The alternative is to use the `imagebase` parameter with the value of `codebase` as the location for the icon images used by applications. The `codebase` value refers to the `forms/java` directory, which contains all of the Forms client Java archive files. For performance reasons, it is not a good idea to store images here.

Instead, you should bundle the icons into a separate archive file, which improves performance because archives are cached permanently on the client. Follow these steps to create this archive file.

1. Verify that the `jar` command succeeds. If it does not, then you need to ensure that there is a JDK installed on your system with a correct PATH environment variable entry (pointing to the `JDK_HOME/bin` directory).
2. Navigate to the directory containing the application images and issue the command:

   `jar -cvf <application>_images.jar *.<extension>`

   where:

   - *`application`* is the name of the application
   - *`extension`* is the extension of the image file (for example,.gif)

   A jar file, `<application>_images.jar`, is created in the current directory.
3. Copy `<application>_images.jar` to the `forms/java` directory.
4. Edit the `formsweb.cfg` file, adding the `imageBase=codebase` parameter to the custom application section for the application.
5. Add the `<application>_images.jar` file to the archive path used by the application by adding the following line to the custom application section:

   ```
   archive=frmall.jar,<application>_images.jar
   ```

See Section 4.7, "Deploying Fonts, Icons, and Images Used by Forms Services" for more information on deploying custom icon files with Oracle Forms Services.

13.2.8.2 Upgrading Integrated Calls to Oracle Forms to use Oracle Reports

Integrated calls to Oracle Reports in Forms are no longer handled by a client-side background engine. Oracle Forms Services requires that applications use the `RUN_REPORT_OBJECT` built-in, calling Oracle Reports to process integrated reports. Oracle Reports is set up as part of the Business Intelligence and Forms installation.

Follow these steps to upgrade the call:

1. Change all occurrences of `RUN_PRODUCT (Reports,...)` to the equivalent call using `RUN_REPORT_OBJECT()`.
2. Add the location of the application's Reports modules to use the `Reports_Path` of Oracle Reports.
3. Change `RUN_REPORT_OBJECT` to reference Oracle Reports.

   For more information, see *Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services*.

13.2.8.3 Creating Forms Listener Servlet Alias Names

In Forms Services Release 6*i*, before patch 8, it was necessary to create alias names for the Forms servlet in the `$ORACLE_HOME/Apache/Apache/JServ/conf/zone.properties` file in order to use individual environment files for different applications. The Forms servlet in Oracle Forms Services does not require this. You can set the environment file name in the `formsweb.cfg` file using the `envFile` parameter, shown below:

```
envFile=myApp.env
```

Alias names for the Forms servlet are no longer created in `$ORACLE_HOME/Apache/Apache/JServ/conf/zone.properties`. Instead, they are created in `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF/web.xml`.

To create the alias names, copy the content between the `<servlet>` and `</servlet>` tags and change the servlet's name. To create a URL mapping for the new servlet alias name, add the following to the file:

```
<servlet-mapping>
<servlet-name>new servlet name</servlet-name>
<url-pattern>/new url name*</url-pattern>
</servlet-mapping>
```

13.2.8.4 Accessing the Listener Servlet Administration Page

You can display a test page for the Listener Servlet in Oracle9*i*AS Forms Services Release 6*i* by accessing the following URL:

`http://<hostname>:<port>/servlet/oracle.forms.servlet.ListenerServlet`

The information displayed depends on the value of the initialization parameter `TestMode`. This parameter is set in the `<source_OH>/Apache/Apache/JServ/conf/zone.properties` file.

You can display the test page for Oracle Forms Services with the following URL:

`http://<hostname>:<port>/forms/frmservlet/admin`

The information displayed depends on the value of the initialization parameter `TestMode`. This parameter is set in the `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF/web.xml` file. An example is shown below:

```
<init-param>
<!-- Display sensitive options on the /admin page ? -->
    <param-name>TestMode</param-name>
    <param-value>true</param-value>
</init-param>
```

13.3 Validating the Oracle Forms Services Upgrade

After you complete the upgrade tasks, ensure that the upgraded version of the Oracle Forms Services is working as expected. You must devise and perform specific tests for applications and configuration elements that are unique to your site. Compare the performance and characteristics of each application in the source and destination installations.

In Oracle9iAS Release 1 (1.0.2.2.x), the forms application URL is typically:

```
http://<hostname>:<port>/servlet/<forms servlet alias>?<forms
application name>
```

In Oracle Forms 11*g*, the forms application URL is typically:

```
http://<hostname>:<port>/forms/<forms servlet alias>?form=<forms
application name>
```

14 Performance Tuning Considerations

This chapter contains the following sections:

Tuning the connection between Oracle Forms Services and the Oracle Database Server is beyond the scope of this chapter.

14.1 Built-in Optimization Features of Forms Services

The Oracle Forms Services and Java client include several optimizations that fit broadly into the following categories:

14.1.1 Monitoring Forms Services

Use Fusion Middleware Control to monitor Oracle Forms Services and review metrics information, including:

- Forms Services Instances
- Events
- User Sessions
- Forms Trace

14.1.1.1 Monitoring Forms Services Instances

Use the Forms Home page to monitor metrics for a Forms Services instance.

1. Start Enterprise Manager Fusion Middleware Control.

2. From the Enterprise Manager Fusion Middleware Control main page, select the link to the Forms Services instance that you want to monitor.

   The Forms Home page for the Forms Services instance displays the following:

   - Status of Forms application instance (up, down, unknown)
   - URL of the Forms Services instance being monitored
   - Number of Forms sessions

   Additionally, you can navigate to the following detail pages:

   - Performance Summary
   - Servlet Logs
   - Session Details
   - Web Configuration
   - Environment Configuration
   - Trace Configuration
   - User Sessions
   - JVM Configuration
   - JVM Controllers

     In the Performance Summary page, you can add charts for other Forms metrics to the page dynamically by using the Show Metric Palette. You can also overlay metrics to compare them. For example, drag and drop Private Memory consumed by two JVM Controllers into one chart to compare them. For more information, see the Oracle Fusion Middleware Performance Guide.

14.1.1.2 Monitoring Forms Events

Use the Enterprise Manager Fusion Middleware Control to enable tracing for all events or specific ones. See Table 14–1 for a list of tasks you can perform on this page.

Table 14–1 Monitoring Forms Events

Task	See Section
Monitoring metrics for user sessions	"To view Forms user sessions:"
Sorting metrics information	"To sort the list of Forms user sessions:"
Searching for metrics information	"To search for a Forms user sessions:"

14.1.2 Forms Services Web Runtime Pooling

Forms Runtime Pooling (or Forms Runtime prestart) enables the startup of a configurable number of application runtime engines prior to their usage. Runtime Pooling provides quick connections at server peak times, which shortens the server-side application startup time. Runtime pooling is useful for situations where server configurations have a small window in which many users connect to a Forms application. All prestarted runtime engines run in the same environment serving the same application.

14.1.2.1 Configuring Prestart Parameters

Use Enterprise Manager Fusion Middleware Control to configure runtime pooling for Forms Services with the following parameters as described in Table 14–2:

Table 14–2 Forms Runtime Pooling Parameters

Parameter Name	Data type	Description	Default Value
`prestartRuntimes`	boolean	Runtime pre starting or pooling is enabled only if true	false
`prestartInit`	integer	Number of the runtime processes that should be spawned initially	1
`prestartTimeout`	integer	Time in minutes after which all the prestarted processes of this pool (configuration section) will be stopped. A runtime process is removed from the prestart pool once client connection is made and thus will not be stopped.	0 (When set to zero the timer never starts)
`prestartMin`	integer	Minimum number of runtime processes to exist in the pool.	0
`prestartIncrement`	integer	The number of runtime processes to be created when the number of prestarted runtime processes is less than minRuntimes.	0

Note that `prestartMin` defines the minimum number of pre-started runtimes that must exist at any time while runtime pooling is still active for a specific application. The minimum value must be less than or equal to what's defined for the `prestartInit` parameter. The `prestartMin` parameter can be modified at any time and does not require the application server to be restarted. The new entries will be picked up when a client requests a connection to a pre-started runtime process and the prestarted runtime processes have not timed out. Once they have timed out, an application uses default behavior and a minimum threshold is not maintained.

Each configuration section can specify values for these parameter. If the `prestartRuntimes = true` entry is found, but there is no associating prestart parameter, then default values are used.

In a load balanced system that has multiple instances of Oracle WebLogic Managed Server, the various values provided for the above parameters are on a per JVM basis, and not the total for the application.

14.1.2.2 Starting Runtime Pooling

An Administrator can configure specific application(s), from the Enterprise Manager Fusion Middleware Control, to enable Runtime Pooling. On the startup of the application server (Oracle WebLogic Managed Server), the configured number of Forms Runtime processes are pre-started for each application.

In the initialization phase of the Forms servlet, the configuration file (`formsweb.cfg`) is read and the server pre-starts the applications which have the `prestartRuntimes` parameter enabled.

14.1.3 Minimizing Client Resource Requirements

The Java client is primarily responsible for rendering the application display. It has no embedded application logic. Once loaded, a Java client can display multiple forms simultaneously. Using a generic Java client for all Oracle Forms applications requires fewer resources on the client when compared to having a customized Java client for each application.

The Java client is structured around many Java classes. These classes are grouped into functional subcomponents, such as displaying the splash screen, communicating with the network, and changing the look-and-feel. Functional subcomponents allow the Forms Developer and the Java Virtual Machine (JVM) to load functionality as it is needed, rather than downloading all of the functionality classes at once.

14.1.4 Minimizing Forms Services Resource Requirements

When a form definition is loaded from an FMX file, the profile of the executing process can be summarized as:

- Encoded Program Units
- Boilerplate Objects/Images
- Data Segments

Of these, only the data segments section is unique to a given instance of an application. The encoded program units and boilerplate objects/images are common to all application users. Forms Services maps the shared components into physical memory, and then shares them between all processes accessing the same FMX file.

The first user to load a given FMX file will use the full memory requirement for that form. However, subsequent users will have a greatly reduced memory requirement, which is dependent only on the extent of local data. This method of mapping shared components reduces the average memory required per user for a given application.

14.1.5 Minimizing Network Usage

Bandwidth is a valuable resource, and the general growth of Internet computing puts an ever increasing strain on the infrastructure. Therefore, it is critical that applications use the network's capacity sparingly.

Oracle Forms Services communicates with the Java client using metadata messages. Metadata messages are a collection of name-value pairs that tell the client which object to act upon and how. By sending only parameters to generic objects on the Java client,

there is approximately 90-percent less traffic (when compared to sending new code to achieve the same effect).

Oracle Forms Services intelligently condenses the data stream in three ways:

- When sets of similar messages (collections of name-value pairs) are sent, the second and subsequent messages include only the differences from the previous message. This results in significant reductions in network traffic. This process is called *message diff-ing*.
- When the same string is to be repeated on the client display (for example, when displaying multiple rows of data with the same company name), Oracle Forms Services sends the string only once, and then references the string in subsequent messages. Passing strings by reference increases bandwidth efficiency.
- Data types are transmitted in the lowest number of bytes required for their value.

14.1.6 Maximizing the Efficiency of Packets Sent Over the Network

The extensive use of triggers within the Forms Developer model is a strength, but they can increase the effect of latency by requiring a network round trip for each trigger. Latency can be the most significant factor that influences the responsiveness of an application. Note that latency is not the same as network speed. Network speed involves a measure of the bits that can be transported per time unit whereas latency is the time taken for one bit to travel from one end-point to the other. One of the best ways to reduce the effects of latency is to minimize the number of network packets sent during a conversation between the Java client and the Forms Services.

Oracle Forms Services implements event bundling by grouping trigger events together through Event Bundling. Event Bundling gathers all of the events triggered while navigating between the two objects, and delivers them as a single packet to Oracle Forms Services for processing.

For example, when a user navigates from item A to item B (such as when tabbing from one entry field to another), a range of pre- and post-triggers may fire, each of which requires processing on the Forms Services. When navigation involves traversing many objects (such as when a mouse click is on a distant object), Event Bundling gathers all events from all of the objects that were traversed, and delivers the group to Oracle Forms Services as a single network message.

14.1.7 Rendering Application Displays Efficiently on the Client

All boilerplate objects in a given form are part of a Virtual Graphics System (VGS) tree. VGS is the graphical subcomponent that is common to all Forms Developer products. VGS tree objects are described using attributes such as coordinates, colors, line width, and font. When sending a VGS tree for an object to the Java client, the only attributes that are sent are those that differ from the defaults for the given object type.

Images are transmitted and stored as compressed JPEG images. This reduces both network overhead and client memory requirements.

Minimizing resources includes minimizing the memory overhead of the client and server processes. Optimal use of the network requires that bandwidth be kept to a minimum and that the number of packets used to communicate between the client and Oracle Forms Services be minimized in order to contain the latency effects of the network.

14.2 Tuning Oracle Forms Services Applications

An application developer can take steps to ensure that maximum benefits are gained from Forms Services' built-in architectural optimizations. The remainder of this chapter discusses key performance issues that affect many applications and how developers can improve performance by tuning applications to exploit Forms Services features.

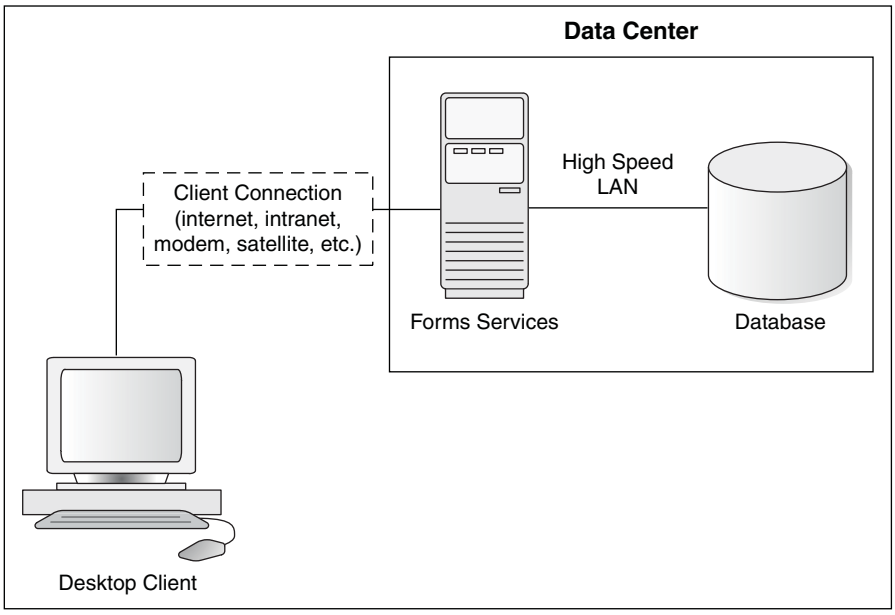
14.2.1 Location of the Oracle Forms Services with Respect to the Data Server

The Forms Java client is only responsible to display the GUI objects. All of the Oracle Forms logic runs in Oracle Forms Services, on the middle tier. This includes inserting or updating the data to the database, querying data from the database, executing stored procedures on the database, and so on. Therefore, it is important to have a high-speed connection (high bandwidth and not low latency) between the application server and the database server.

All of this interaction takes place without any communication to the Forms Java client. Only when there is a change on the screen is there any traffic between the client and Forms Services. This allows Oracle Forms applications to run across slower networks (high latency networks), such as with modems or satellites.

The configuration in Figure 14–1, displays how Forms Services and the database server are co-located in a data center.

Figure 14–1 Co-Locating the OracleAS Forms Services and Database Server



14.2.2 Minimizing the Application Startup Time

First impressions are important, and a key criterion for any user is the time it takes to load an application. Startup time is regarded as overhead. It also sets an expectation of future performance. When a business uses thin-client technologies, the required additional overhead of loading client code may have a negative impact on users. Therefore, it is important to minimize load time wherever possible.

After requesting an Oracle Forms application, several steps must be completed before the application is ready for use:

1. Invoke Java Virtual Machine (JVM).
2. Load all initial Java client classes, and authenticate security of classes.
3. Display splash screen.
4. Initialize form:
   a. Load additional Java classes, as required.
   b. Authenticate security of classes.
   c. Render boilerplate objects and images.
   d. Render all elements on the initial screen.
5. Remove splash screen.
6. Form is ready for use.

An application developer has little influence on the time it takes to launch the JVM. However, the Java deployment model and the structure of the Oracle Forms Developer Java client allow the developer to decide which Java classes to load and how. This, in turn, minimizes the load time required for Java classes.

The Java client requires a core set of classes for basic functionality (such as opening a window) and additional classes for specific display objects (such as LOV items). These classes must initially reside on the server, but the following techniques can be used to improve the time it takes to load these classes into the client's JVM:

- Using Java Files
- Using Caching

14.2.2.1 Using Java Files

Java provides the Java Archive (Jar) mechanism to create files that allow classes to be grouped together and then compressed (zipped) for efficient delivery across the network to the client. Once used on the client, the files are cached for future use.

It is also possible to double jar a file. This saves about 700k when done with `frmall.jar`. For Oracle's plugin, the resulting file must have a suffix of `jarjar`.

The following sections describe the pre-configured Jar files that Oracle Forms Services provides to support typical deployment scenarios.

14.2.2.2 Using Oracle's Java Plug-in

frmall.jar includes all required classes for running with the Java Plug-in.

To specify one or more Jar files, use the archive setting in the named configuration section of the Forms Configuration file (`formsweb.cfg`). For example,

```
[MyApp]
archive=frmall.jar
```

14.2.2.3 Using Caching

Oracle's Java Plug-in supports the caching of Jar files for Oracle Forms Services. When the JVM references a class, it first checks the local client cache to see if the class exists in a pre-cached Jar file. If the class exists in cache, JVM checks the server to see if there

is a more current version of the Jar file. If there isn't, the class is loaded from the local cache rather than from across the network.

Be sure that the cache is of proper size to maximize its effectiveness. Too small a cache size may cause valid Jar files to be overwritten, thereby requiring that another Jar file be downloaded when the application is run again. The default cache size is 20MB. This size should be compared with the size of the cache contents after successfully running the application.

Jar files are cached relative to the host from which they were loaded. This has implications in a load-balancing architecture where identical Jar files from different servers can fill the cache. By having Jar files in a central location and by having them referenced for each server in the load-balancing configuration, the developer can ensure that only one copy of each Jar file is maintained in the client's cache. A consequence of this technique is that certain classes within the Jar file must be signed to enable connections back to servers other than the one from which they were loaded. The Oracle-supplied Jar files already pre-sign the classes.

14.2.3 Reducing the Required Network Bandwidth

The developer can design the application to maximize the data stream compression, called message-diffing, that Forms automatically performs. This means that forms sends along data stream compression by using message diff-ing, which sends along only the information that differs from one message to another. The following steps can be taken to reduce the differences between messages:

- **Promote similarities between objects.** Using similar objects improves *message diff-ing* effectiveness (in addition to being more visually appealing to the user). The following steps encourage consistency between objects:
  - Accept default values for properties, and change only those attributes needed for the object.
  - Use Smart Classes to describe groups of objects.
  - Lock the look-and-feel into a small number of visual attributes.
- **Reduce the use of boilerplate text.** As a developer, you should use the PROMPT item property rather than boilerplate text wherever applicable. Forms Developer 6.0 and higher includes the Associate Prompt feature, which allows boilerplate text to be re-designated as the prompt for a given item.
- **Reduce the use of boilerplate items (such as arcs, circles, and polygons).** All boilerplate items for a given Form are loaded at Form initialization. Boilerplate items take time to load and use resources on the client whether they are displayed or not. Common boilerplate items, namely rectangles and lines, are optimized. Therefore, restricting the application to these basic boilerplate items reduces network bandwidth and client resources while improving startup times.
- **Keep navigation to a minimum.** An Event Bundle is sent each time a navigation event finishes, whether the navigation extends over two objects or many more. Design Forms that do not require the user to navigate through fields when default values are being accepted. A Form should encourage the user to quickly exit once the Form is complete, which causes all additional navigation events to fire as one Event Bundle.
- **Reduce the time to draw the initial screen.** Once the Java client has loaded the required classes, it must load and initialize all of the objects to be displayed before it can display the initial screen. By keeping the number of items to a minimum, the

initial screen is populated and displayed to the user more promptly. Techniques that reduce the time to draw the initial screen include:

- Providing a login screen for the application with a restricted set of objects (such as a title, small logo, username, and password).
- On the Form's initial display, hiding elements not immediately required. Use the canvas properties:

  `RAISE ON ENTRY = YES` (Canvas only)

  `VISIBLE = NO`

  Pay attention to TAB canvases that consist of several sheets where only one will ever be displayed. For responsive switching between tabs, all items for all sheets on the canvas are loaded, including those that are hidden behind the initial tab. Consequently, the time taken to load and initialize a TAB canvas is related to all objects on the canvas and not just to those initially visible.

  Tip: When using Tab canvases, use stacked canvases and display the right canvas in the when-tab-page-changed trigger. Remember to set the properties `RAISE ON ENTRY = YES` and `VISIBLE = NO` for all the canvases not displayed in the first screen.

- **Disable MENU_BUFFERING.** By default, MENU_BUFFERING is set to True. This means that changes to a menu are buffered for a future "synchronize" event when the altered menu is re-transmitted in full. (Most applications make either many simultaneous changes to a menu or none at all. Therefore, sending the entire menu at once is the most efficient method of updating the menu on the client.) However, a given application may make only minimal changes to a menu. In this case, it may be more efficient to send each change as it happens. You can achieve this using the statement:

  ```
  Set_Application_Property (MENU_BUFFERING, 'false');
  ```

  Menu buffering applies only to the menu properties of LABEL, ICON, VISIBLE, and CHECKED. An ENABLE/DISABLE event is always sent and does not entail the retransmission of an entire menu.

14.2.4 Other Techniques to Improve Performance

The following techniques may further reduce the resources required to execute an application:

- **Examine timers and replace with JavaBeans.** When a timer fires, an asynchronous event is generated. There may not be other events in the queue to bundle with this event. Although a timer is only a few bytes in size, a timer firing every second generates 60 network trips a minute and almost 30,000 packets in a typical working day. Many timers are used to provide clocks or animation. Replace these components with self-contained JavaBeans that achieve the same effect without requiring the intervention of Forms Services and the network.
- **Consider localizing the validation of input items.** It is common practice to process input to an item using a When-Validate-Item trigger. The trigger itself is processed on the Forms Services. You should consider using pluggable Java components to replace the default functionality of standard client items, such as text boxes. Then, validation of items, such as date or max/min values, are contained within the item. This technique opens up opportunities for more

complex, application-specific validation like automatic formatting of input, such as telephone numbers with the format (XXX) XXX-XXXX.

- **Reduce the application to many smaller forms, rather than one large form.** By providing a fine-grained application, the user's navigation defines which objects are loaded and initialized from the Forms Services. With large Forms, the danger is that the application is delayed while objects are initialized, many of which may never be referenced. When chaining Forms together, consider using the built-ins OPEN_FORM and NEW_FORM:
  - With OPEN_FORM, the calling Form is left open on the client and the server, so that the additional Form on both the client and the server consumes more memory. However, if the Form is already in use by another user, then the increase in server memory is limited to just the data segments. When the user returns to the initial Form, it already resides in local memory and requires no additional network traffic to redisplay.
  - With NEW_FORM, the calling Form is closed on the client and the server, and all object properties are destroyed. Consequently, it consumes less memory on the server and client. Returning to the initial Form requires that it be downloaded again to the client, which requires network resources and startup time delays. Use OPEN_FORM to display the next Form in an application unless it is unlikely that the initial form will be called again (such as a login form).
- **Avoid unnecessary graphics and images.** Wherever possible, reduce the number of image items and background images displayed in your applications. Each time an image is displayed to application users, the image must be downloaded from the application server to the user's Web browser. To display a company logo with your Web application, include the image in the HTML file that downloads at application startup. Do this instead of including it as a background image in the application. As a background image, it must be retrieved from the database or file system and downloaded repeatedly to users' computers.

14.3 Web Cache and Forms Integration

Oracle Web Cache can be used as a load balancer with Oracle Forms applications. A Forms client needs to be able to communicate with the same instance of the server process for the duration of a session. Since Forms applications are stateful, Web Cache must be configured for stateful load balancing using its session binding feature.

Figure 14–2 Web Cache Load Balancing

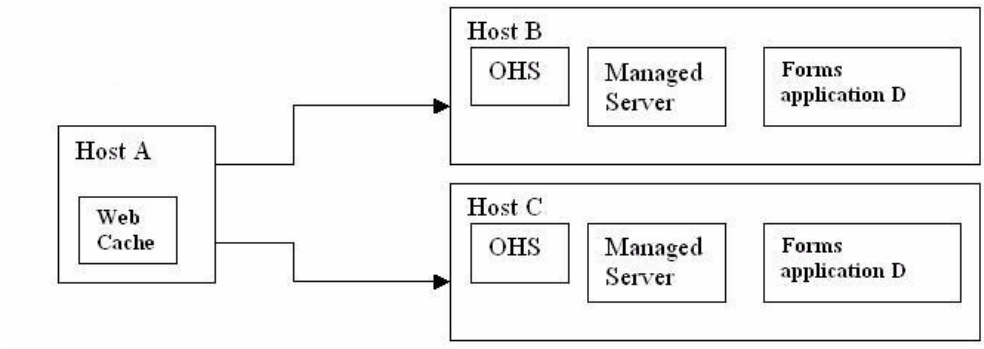


Figure 14–2 assumes a setup where a single Web Cache instance is load balancing two Application Server tiers. This can be described as following:

1. Oracle Web Cache instance running on Host A
2. Oracle HTTP Server instance and Oracle WebLogic Managed Server on Host B running Oracle Forms application D
3. Oracle HTTP Server instance and Oracle WebLogic Managed Server on Host C running Oracle Forms application D

> **Note:** There can be more Oracle HTTP Server/Oracle WebLogic Managed Server, but only two instance pairs are described here for purpose of simplification. The Oracle HTTP Server/Oracle WebLogic Managed Server is not configured for clustering/active failover because Oracle Forms applications cannot take advantage of Oracle WebLogic Server active failover.

Prerequisites

1. Modify the Forms J2EE application deployment descriptor (`weblogic.xml`) to override the default session tracking entries and to enable the cookies for session tracking. For more information, refer to Section 5.2.4, "Modification of Forms J2EE Application Deployment Descriptors".

   Modify the deployment plan. The following is a sample of the deployment plan with the added entries highlighted in bold:

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/weblogic/deployment-plan
http://xmlns.oracle.com/weblogic/deployment-plan/1.0/deployment-plan.xsd"
global-variables="false">
  <application-name>formsapp</application-name>
  <variable-definition>
    <variable>
      <name>vd-/scratch/t_work/Oracle/Middleware/as_1/forms</name>
      <value>/scratch/t_work/Oracle/Middleware/as_1/forms</value>
    </variable>
    <variable>
      <name>vd-/scratch/t_work/Oracle/Middleware/user_
projects/domains/ClassicDomain/config/fmwconfig/servers/WLS_
FORMS/applications/formsapp_11.1.1/config/forms</name>
      <value>/scratch/t_work/Oracle/Middleware/user_
projects/domains/ClassicDomain/config/fmwconfig/servers/WLS_
FORMS/applications/formsapp_11.1.1/config/forms</value>
    </variable>
    <variable>
       <name>Frmapp_url-rewriting-enabled_variable</name>
       <value>false</value>
    </variable>
    <variable>
       <name>Frmapp_cookies-enabled_variable</name>
       <value>true</value>
    </variable>
    <variable>
       <name>Frmapp_cookie-http-only_variable</name>
       <value>false</value>
    </variable>
```

```
  </variable-definition>
  <module-override>
    <module-name>formsapp.ear</module-name>
    <module-type>ear</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-application</root-element>
      <uri>META-INF/weblogic-application.xml</uri>
    </module-descriptor>
    <module-descriptor external="false">
      <root-element>application</root-element>
      <uri>META-INF/application.xml</uri>
    </module-descriptor>
    <module-descriptor external="true">
      <root-element>wldf-resource</root-element>
      <uri>META-INF/weblogic-diagnostics.xml</uri>
    </module-descriptor>
  </module-override>
  <module-override>
    <module-name>formsweb.war</module-name>
    <module-type>war</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-web-app</root-element>
      <uri>WEB-INF/weblogic.xml</uri>
      <variable-assignment>
        <name>vd-/scratch/t_work/Oracle/Middleware/as_1/forms</name>
<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="java/*"]/local
-path</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>vd-/scratch/t_work/Oracle/Middleware/as_1/forms</name>
<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="webutil/*"]/lo
cal-path</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>vd-/scratch/t_work/Oracle/Middleware/user_
projects/domains/ClassicDomain/config/fmwconfig/servers/WLS_
FORMS/applications/formsapp_11.1.1/config/forms</name>
<xpath>/weblogic-web-app/virtual-directory-mapping/[url-pattern="registry/*"]/l
ocal-path</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>Frmapp_url-rewriting-enabled_variable</name>
        <xpath>/weblogic-web-app/session-descriptor/url-rewriting-enabled
</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>Frmapp_cookies-enabled_variable</name>
        <xpath>/weblogic-web-app/session-descriptor/cookies-enabled</xpath>
      </variable-assignment>
      <variable-assignment>
        <name>Frmapp_cookie-http-only_variable</name>
       <xpath>/weblogic-web-app/session-descriptor/cookie-http-only</xpath>
      </variable-assignment>
    </module-descriptor>
    <module-descriptor external="false">
       <root-element>web-app</root-element>
       <uri>WEB-INF/web.xml</uri>
    </module-descriptor>
  </module-override>
</deployment-plan>
```

2. Add the virtual host directives to `forms.conf` as shown below:

```
<VirtualHost *:8888>
     ServerName hostA:8090
     UseCanonicalName On
     CookieTracking On
     WLCookieName cookieName
#
# virtual mapping for the /forms/html mapping.
#
RewriteEngine on
RewriteRule ^/forms/html/(..*) /workaroundhtml/$1 [PT]
AliasMatch ^/workaroundhtml/(..*) ". . . . . .
/config/FormsComponent/forms/html/$1"

<Location /forms>
        SetHandler weblogic-handler
        WebLogicCluster    HostB:9001
        DynamicServerList OFF
</Location>
</VirtualHost>
```

> **Note:** The `ServerName` entry must have the Web Cache hostname and port number. The value of the `WLCookieName` entry must be unique across hosts. For example, `cookieHostA`, `cookieHostB`, and so on. The port number 8888 in the `VirtualHost` directive corresponds to the OHS HTTP port. You can use `$ORACLE_INSTANCE/bin/opmnctl status -l` to obtain the actual OHS HTTP port value.

3. The Application Server hosts (Host B and Host C) must have the same FMW patch set version.
4. Ensure that the Forms configuration files are synchronized across the Application Server hosts. This means that you must create matching entries in the Forms configuration files (`formsweb.cfg`, `default.env`, `Registry.dat`, and so on) across all the Application Server hosts.

To Configure Session Binding in Web Cache:

1. Set up Web Cache running on Host A as described in *Configuring Oracle Web Cache as a Software Load Balancer* in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.
2. Add the Application servers, Host B and Host C as Origin Servers as described in *Specify Origin Server Settings* in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

   Ensure to specify the URL as `/forms/lservlet` in the **Ping URL** field.
3. Create ordered mappings of sites to origin servers, Host B and Host C as described in *Map Site Definitions to Origin Servers* in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.
4. Configure session binding in Web Cache as described in *Configuring Session Binding* in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

   Ensure to select the **Cookie based session binding with any Set-Cookie** option.

To test the setup:

1. Using a browser, point it to the Web Cache host and access Oracle Forms application D. Ensure that the application works as expected. Keep the browser window open.
2. Use the OHS access logs to identify the Oracle HTTP Server/Oracle WebLogic Managed Server that handled the requests. For example, assume this is Host B and shut down the Oracle HTTP Server/WebLogic Managed Server on that host. Now only the Oracle HTTP Server/WebLogic Managed Server running on Host C will be accessible.
3. Using the same browser that is running the Oracle Forms client, access Oracle Forms application D again. The request will fail, and the Forms client will lose its session. Note that Oracle Forms session state is not replicated among Oracle WebLogic Managed Server.
4. Next, clear the browser cookies and open a browser window. Point it to the Web Cache host and access Oracle Forms application D. Web Cache will direct the requests to the remaining Oracle HTTP Server/WebLogic Managed Server running on Host C. Ensure that the application works as expected.
5. Restart the Oracle HTTP Server/WebLogic Managed Server on Host B. Using a browser, log on to the Web Cache Manager.
6. In the Oracle Enterprise Manager navigator panel, select **Web Cache** and navigate to the **Home** page.
7. In the **Web Cache** Home page, ensure that Host B listed under Origin Servers is checked.

Note: For information about load balancing scenarios and setting up clustered Web Cache, see *Oracle Fusion Middleware High Availability Guide*. For more information about Web Cache, see *Oracle Fusion Middleware Web Cache Administrator's Guide*.

A

Troubleshooting Oracle Forms Services

This chapter contains the following:

This chapter provides information to help you resolve problems that might occur when you run an application over the Web using Oracle Forms. It contains an outline of common causes for errors, the method you can use to verify your installation, and the tools and techniques provided to diagnose problems.

This chapter is also a subset of the whitepaper *Oracle Forms Diagnostic Techniques* that can be found at `http://www.oracle.com/technology/products/forms/`.

A.1 Verifying The Installation

If there is something wrong with the installation, then it will result in faulty configuration and Oracle Forms will not run correctly. After the Oracle Universal Installer indicates that Fusion Middleware Control was successfully installed, you can verify whether Oracle Forms Services is correctly configured or not. You can use these tools:

Section A.1.1, "Use The Web Form Tester"

Section A.1.2, "Find Port Information"

A.1.1 Use The Web Form Tester

The Web Form Tester is available with your Oracle Fusion Middleware installation. To verify whether the Oracle installation and configuration of Forms Services is correct, run the Web Form Tester on the middle tier. The following is an example of how this can be done on a Windows computer.

1. Start the Admin server for the WebLogic Server domain by selecting **Start | Program Files | Oracle WebLogic Server | User Projects | Domain | Start Admin Server for WLS Domain**, if it is not already started.

2. If the managed server is not up, perform the following steps:
    1. Start the node manager by selecting **Start | Program Files |Oracle WebLogic | WebLogic Server 11gR1 | Tools | Node Manager**, if it is not already started.
    2. Start Forms Services from the WebLogic Administrator Console.
3. Open an instance of the browser by typing `<ORACLE_HOME>/tools/web/html/runform.htm` for the URL and press ENTER. Replace ORACLE_HOME with your actual Oracle home for Oracle Fusion Middleware.
4. Alternatively, you can run the Web Form Tester by selecting **Start | Program Files | <Oracle_Home> | Forms Services | Run a Form on the Web** from the Windows Start menu for Oracle Fusion Middleware.
5. Enter the Web port and click the **Run Form** button. See Section A.1.2, "Find Port Information" to learn how to find out the Web port.
6. If the installation of Oracle Fusion Middleware is correct, you will see a success message in the Web browser. Also, it can be tested from a client computer whether the basic Forms setup in Oracle Fusion Middleware on the middle tier is installed correctly or not by the installer. You can run the test form from any client computer by running it from the browser with the URL `http://example.com:NNNN/forms/frmservlet?form=test.fmx`.

A.1.2 Find Port Information

When in doubt or you need to know what port numbers to use to run Forms after installation, you can look at port information in the file `<ORACLE_HOME>/install/portlist.ini`. Use the appropriate port numbers for your installation.

A.2 Diagnosing FRM-XXXXX Errors

Use these tools to diagnose and resolve FRM-XXXXX errors:

- Section A.2.1, "The Oracle Forms Applet"

A.2.1 The Oracle Forms Applet

The brief message about the FRM error should help in identifying the basic cause of the problem. Often, everything required to identify the cause an FRM error is contained in the error reported by the Forms applet. When a FRM error is raised, the error dialog will have a **Details** button. Pressing the 'Details' button will show the current Java stack. The exact stack is tied to the root cause and the version of Oracle Forms. This is due to the differing package structure used for the applet class files in the different releases.

A.3 Diagnosing Server Crashes with Stack Traces

This section contains the following:

- Section A.3.1, "About Stack Traces"
- Section A.3.2, "Configuring and Using Stack Traces"

If the Forms web runtime terminates unexpectedly, then it writes a stack trace to the directory `$ORACLE_INSTANCE/FormsComponent/forms/trace`. The filename will

have the format <forms_runtime_process>_dump_<process id>.The dump file contains a stack trace of the running process, and shows the last successful operation performed by Forms. This core file can be used to assemble a stack trace with symbol names using GNU Debugger, dbx or similar debugging tool on the machine where the dump occurred.

A.3.1 About Stack Traces

A stack trace is useful for two reasons:

- The information in the stack can be used to identify a known issue. It is not 100% reliable, but an identical stack trace is a good indicator of a matching problem. Even if it is not the same, there may be a workaround or patch for an existing bug that can be tested.
- If the problem is not a known bug, then the stack may provide valuable information to assist development efforts to pinpoint the cause.

A.3.2 Configuring and Using Stack Traces

This section contains the following:

- Section A.3.2.1, "Verifying the Environment"
- Section A.3.2.2, "Understanding UNIX Stack Traces"
- Section A.3.2.3, "Understanding Windows Stack Traces"

A.3.2.1 Verifying the Environment

In order to test stack tracing on UNIX or Windows you can set the environment variable FORMS_DELIBERATECRASH. As the name suggests, setting this will cause the forms runtime process to crash. Oracle Forms currently recognizes two settings: 1 and 2. If FORMS_DELIBERATECRASH is set to 1 then forms will crash at runtime whenever the BELL Built-in is executed. If it is set to 2 then forms will crash at runtime whenever a when-button-pressed trigger is fired. This environment variable can be set in the environment (for example, default.env) file.

A.3.2.2 Understanding UNIX Stack Traces

In a UNIX stack trace, the top two functions siehjmpterm() and sigacthandler() are the signal handling code - these functions will often be present in the stack trace. To see the function the program was in when the error occurred you need to read further down the stack.

If you set FORMS_CATCHTERM=0 the two functions do not show up in the dump file. The stack trace is displayed without the crash handling symbols.

A.3.2.3 Understanding Windows Stack Traces

Stack tracing works differently on UNIX and on Windows. The symbol information is contained inside the executable files and shared libraries on Unix. On Windows this information is stripped out at link time and is in the form of binary .sym files. There should be one .sym file for every Oracle Forms executable or DLL. The .sym files are installed by default. On Windows the files are located in the ORACLE_HOME\bin directory. The mechanism on Windows platforms is such that in the event of a crash the Forms runtime process reads all the .sym files that correspond to the forms executable files loaded into memory. It then uses the information in the .sym files to lookup the symbol name.

A.4 Diagnosing Client Crashes

This section contains the following:

A.4.1 About Diagnosing Client Crashes

If the Forms applet disappears unexpectedly, accompanied by a dialog indicating a fatal error, then the Forms applet has crashed. On Windows, a crash will result in the operating system raising an 'illegal operation' dialog, or may cause the "Not responding" flag in Task Manager.To verify the crash, check for a stack trace file on the client. If the client has crashed then a file with the .rpt extension will be created in the same directory as the executable. The root of the filename will be the name of the executable.

Sometimes the applet may appear to have crashed, but no corresponding `.rpt` file can be found. In this case it is likely that the Oracle Forms has unexpectedly disconnected from the client. The applet will still be running, but it has shutdown all the Forms windows, giving the appearance of a client crash.

A.4.2 Diagnosing Hanging Applications

If the client appears to hang then it is important to verify that the server process is still alive. If the server process has not crashed, but the client no longer appears to respond to user interaction then the application is said to be hanging.

In such cases a thread dump can point to the deadlock. A thread dump can be obtained by pressing `t` in the Java console. This displays a list of all the threads running in the client JVM.

The information contained in the dump file is extremely useful to Oracle development, and should be included in any bug filed to report the problem.

A.4.2.1 Causes of Hanging Applications

One cause could be a mismatch between the Java class files and the Oracle Forms version. Communication between the applet and the Forms runtime process is based on message ID. If these message ID's are out of sync, then the applet may not understand an instruction from the server, and vice versa. If you are using Jar files, then try with the <ARCHIVE> tag removed. If the problem persists then pull the correct class files off the installation/patch CD by hand.

Another cause is that the Forms Runtime process may have died. Check if the Forms Runtime process on the server is still alive. Check that the `FORMS_TIMEOUT` parameter is set. It defines how long the server should wait for a ping from the Oracle Forms client, only cleaning up the runtime process when there has been no activity from the Forms client for the specified time. The client sends out a `HEARTBEAT` every two minutes by default. If FORMS_TIMEOUT is set to two minutes or longer, the server will stay up as long as it hears a HEARTBEAT from the client. Set to shorter than the HEARTBEAT interval, it will shut down after the interval specified in `FORMS_TIMEOUT`. You can set the interval by setting the `HEARTBEAT` applet parameter in `formsweb.cfg`. For more information, see Section 8.6.3, "Configuring Asynchronous Communication." Although this is primarily intended to prevent orphaned server processes, it can also prevent the unwanted premature cleanup of server processes.

A.5 Forms Trace and Servlet Logging Tools

Forms Trace and Servlet Logging are two more tools to use in troubleshooting your Oracle Forms Environment. For more information on configuring and using Forms Trace, see Chapter 12.1, "About Forms Trace" and Chapter 12.6, "Taking Advantage of Oracle Diagnostics and Logging Tools".

A.6 Resolving Memory Problems

This section contains the following:

- Section A.6.1, "How Java Uses Memory"
- Section A.6.2, "Setting the Initial Java Heap"
- Section A.6.3, "About Memory Leaks"
- Section A.6.4, "Improving Performance with Caching"

A.6.1 How Java Uses Memory

Like all software programs, a Java applet uses memory. For Java, the language specification requires a 'garbage collector', which is in an internal memory manager for the Java Virtual Machine (JVM). When a Java program needs memory, it requests this memory from the JVM. If there is no memory left, then the JVM will attempt to free some memory by using the garbage collector. The garbage collector will try to release memory that is no longer required to run the program back to the JVM. If there is still insufficient memory to perform the required task then the JVM will attempt to get more memory from the operating system. If that memory allocation fails, then the Java program will be unable to continue.

A.6.2 Setting the Initial Java Heap

You can specify the initial Java Heap (the memory used by the JVM) for your application through Fusion Middleware Control. For the client, you can change the setting in the Java control panel after you've installed the Oracle Java Plug-in.

Note: The JVM will only use the memory it is told it is allowed to use. Even if you have memory available with the operating system, the JVM will not use it if told not to.

A.6.3 About Memory Leaks

A *memory leak* is an error in a program's dynamic-store allocation logic that causes it to fail to reclaim discarded memory, leading to eventual collapse due to memory exhaustion.

For example, when a program runs it may need to allocate some memory to perform a particular task. If the program has finished with that memory and no longer has any use for it, but fails to make that memory available to other programs running on the computer, then it is said to have leaked the memory.

A typical method used to spot memory leaks is to repeat a series of steps, and observe the memory in use by the application - if the memory usage continues to rise with each iteration, then the assumption is often that the program has a memory leak.

However, some complex applications may choose to retain control of memory it has previously allocated so that it can reuse it at a later point - memory allocation can be

an expensive operation, and if the program expects that it will need more memory later it may be more efficient to keep the unused memory available for reuse.

A.6.3.1 Memory Leaks in Java

The Java language specification demands that the JVM has a garbage collector. In Java, the programmer allocates memory by creating a new object. There is no way to de-allocate that memory. Periodically the garbage collector sweeps through the memory allocated to the program, and determines which objects it can safely destroy, therefore releasing the memory. To determine which objects it can safely destroy, the garbage collector uses a 'mark and sweep' algorithm. The garbage collector scans the dynamically allocated memory for objects, marking those which still have active references to them.

After all possible paths to objects have been investigated, unmarked objects that are known to be no longer needed can be garbage collected. A common myth with Java programming is that the presence of a garbage collector means that there can be no memory leaks. This is not true because the garbage collector simply marks those objects, which have active references, and destroys those that do not. It is possible to have an active reference to an object that is no longer needed. This is a memory leak in Java. The solution to the leak is to destroy the references to the object once it is no longer needed so that the garbage collector can identify it as safe to destroy. If a memory leak exists in a Java program, then calling the garbage collector more frequently will not help.

To complicate matters further, the JVM may choose not to release unused memory back to the operating system. In the real world this seldom matters, as most programs will typically require more memory at some point in the near future and can reuse the free memory in the JVM. However, it is worth bearing in mind that not all the memory allocated to the JVM will be in use by the program running in the JVM.

A.6.3.2 Identifying Memory Leaks

Typically, if a growth in memory usage is observed each time a particular series of operations is performed, then it is a memory leak. The ideal proof is to:

1. Get the form into an initial base state, and record the memory usage,
2. Perform a series of steps to illustrate the problem,
3. Return to the initial base state, and record the memory usage.

By repeating steps 2 and 3, it is possible to determine whether there is a steady memory leak or not. If the growth in memory is small over a large number of iterations, then it may not be a leak at all; it could be that the JVM is retaining unused memory, or the garbage collector is not activating as frequently as expected.

A.6.4 Improving Performance with Caching

When any Java program runs, the Java Virtual Machine needs to load class files. When running over the Internet, the time taken to download a class file each time the program runs can lead to performance problems. In order to solve this download problem, the JDK supports Java Archive (Jar) files. A Jar file is simply a collection of class files bundled into one compressed file. Typically, the size of the Jar file will be much smaller than the combined size of the class files it contains.

When the JVM first references a class, it checks the local computer to see if any of the previously cached Jar files contain this class. If the class does exist in one of the pre-cached Jar files, then the JVM checks to see if there is a newer version of this Jar file on the application server. If there is a newer Jar file available then the new copy of

the Jar file is downloaded to the client cache. If the cached Jar file is up to date, then the class file is loaded from the cached Jar file rather than from over the network.

Caching is important because if the application Jar files do not change, then after the application has run once, and all the Jar files required have been cached on the client, then subsequent invocations of the application will always load the classes from the local cached copies. This can lead to significant performance improvements in the startup time for the application. If new classes are needed to run a specific part of the application, these will be downloaded as required.

A.7 Troubleshooting Tips

The following troubleshooting list will help you deal with complex issues, but it is not a definitive guide to problem solving or a guaranteed set of solutions to your Oracle Forms environment.

Be methodical

Do not immediately leap to the area you believe to be the cause based on a hunch, or a guess - make sure you eliminate the other possibilities first. An easy trap to fall into is that of spending long periods of time trying to find evidence to support your theory, rather than concentrating on what the evidence shows. Do not overlook the trivial or the obvious.

Divide the problem into sections

- Chop the problem into manageable sections - this helps eliminate whole areas from investigation. As you investigate an area and satisfy yourself that the problem does not lie there, you can proceed to the next section. An approach to diagnosing a problem that is often successful is to reduce it to its essential parts. This will be important if you need to discuss the problem with Oracle Support Services to obtain a solution.
- Define what happens, when it happens, how often it happens. Of equal importance is, understanding what does not happen, when it does not happen etc. For example, if a group of users in the same building all get the problem, and it always happens between 9 and 10am, it is just as important to know that it never reproduces in another building, or after 10pm. Perhaps the users only use a particular Form between 9 and 10, or the load on the system is highest between 9 and 10am.

Read the error messages.

It sounds obvious, but often the solution information is within the error text. This document will help you understand the error messages, and help identify what action to take.

Make sure you can reproduce the problem, if possible

If you can reproduce the problem yourself, you may notice some behavior that the end user never spotted - perhaps it had always happened, so they simply assumed it was meant to happen. If you can reproduce the problem then you have already started the first step to resolve it.

Make sure you understand the tools you are trying to use

If you decide to use a diagnostic tool, make sure you know how to use it, and how to interpret the data it produces. Time spent in investigating the usage of a tool before the problem happens is time well invested. Make time to learn the tool as well.

A.8 Need More Help?

You can find more solutions on My Oracle Support (formerly Oracle*MetaLink*) at http://support.oracle.com. If you do not find a solution for your problem, log a service request.

See Also:

- *Oracle Fusion Middleware Release Notes*, available on the Oracle Technology Network: http://www.oracle.com/technology/products/forms/index.html

B

Configuring Java Plug-ins

This section describes the use of Oracle's Java Plug-in as a Web browser plug-in. Oracle Java Plug-in enables users to run Oracle Forms applications using Mozilla Firefox or Internet Explorer. It provides the ability to specify the use of a specific Java Virtual Machine (JVM) on the client. For more information, see the white paper "Using Sun's Java Plug-in" at
http://www.oracle.com/technology/products/forms/index.html.

B.1 Supported Configurations

Oracle supports the Java Plug-in. For more information, see the Java Plug-in Documentation at
http://java.sun.com/products/plugin/reference/docs/index.html.

B.2 Legacy Lifecycle Behavior And Configuration Requirements

In JDK 1.4.1 and later, the Java Plug-in supports the `LEGACY_LIFECYCLE` applet parameter. When this parameter is set to true, a running applet is not destroyed when the user navigates away from a page. Furthermore, when the user navigates back to the page, the running applet is resumed unless:

- The browser must re-issue the request for the applet definition, and
- The response to that request produces an applet definition that differs from the applet definition that was returned by the original request.

B.2.1 Configuration Requirements

To use the `LEGACY_LIFECYCLE` feature for certain configurations, add `LEGACY_LIFECYCLE=true` parameter to the relevant configuration sections, such as in `formsweb.cfg`.

Alternatively, `legacy_lifecycle=true` can be specified on the URL that is used to launch a Forms application. This technique is useful primarily during application development.

In addition, JavaScript must be enabled in the browser from which the Forms application (that specifies `legacy_lifecycle=true`) is launched.

The HTML files must also adhere to certain guidelines. The base HTML files that are shipped with the product already adhere to the required guidelines. However, users who write their own base HTML files must ensure that such files adhere to the following guidelines:

1. The base HTML file must define the serverURL attribute to the value of the `serverURL` variable (`serverURL="%serverURL%"`), in the COMMENT node that has the ID `forms_plugin_info`.
2. The base HTML file must define the `serverURL` applet parameter, and its value must be the value of the `appletServerURL` variable. (Prior to Forms 11*g*, it was set to the value of the `serverURL` variable). This can be accomplished by including

   ```
   <PARAM NAME="serverURL" VALUE="%appletServerURL%">
   ```

   and

   ```
   serverURL="%appletServerURL%"
   ```

   in the OBJECT definition and the EMBED comment in user-written base HTML files. Note that the `appletServerURL` variable should not be set in a configuration file. (If it is, the value is ignored.) Instead, Forms computes its value automatically: if `legacy_lifecycle=true` (in the configuration file or in the initial URL), then the `appletServerURL` variable evaluates to "?", which causes Forms to look for the `serverURL` attribute of the COMMENT node (see above). Otherwise, the `appletServerURL` evaluates to the value of the serverURL variable.
3. The base HTML file must define the `legacy_lifecycle` applet parameter, and the value must not be hard-coded: it must match the value of the `legacy_lifecycle` variable. That is because in Forms 11*g*, the variable also affects the value of the `appletServerURL` variable (as explained above). This can be accomplished by including

   ```
   <PARAM NAME="legacy_lifecycle" VALUE="%legacy_lifecycle%">
   ```

   and

   ```
   legacy_lifecycle="%legacy_lifecycle%"
   ```

   in the OBJECT definition and the EMBED comment in user-written base HTML files.

C

Locations and Samples of Configuration Files

This section includes a list of configuration files and their default locations. This section also includes samples of the default configuration files that are installed on the system. Some default values such as locations and paths may vary.

C.1 Locations of Forms Configuration Files

Table C–1 lists the default locations of Forms configuration files on UNIX. The location of these files in Windows is similar.

Table C–1 List of Files and their Locations in Release 11.1.1.1.0 and 11.1.1.2.0

File Name	Location in Release 11.1.1.1.0	Location in Release 11.1.1.2.0 and later
formsweb.cfg	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/config	$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config
default.env	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/config	$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config

Table C–1 (Cont.) List of Files and their Locations in Release 11.1.1.1.0 and 11.1.1.2.0

File Name	Location in Release 11.1.1.1.0	Location in Release 11.1.1.2.0 and later
base.htm	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/config	$ORACLE_INSTANCE/config/FormsComponent/forms/server
basejpi.htm	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/config	$ORACLE_INSTANCE/config/FormsComponent/forms/server
webutilbase.htm	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/config	$ORACLE_INSTANCE/config/FormsComponent/forms/server
webutiljpi.htm	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/config	$ORACLE_INSTANCE/config/FormsComponent/forms/server
ftrace.cfg	$ORACLE_INSTANCE/config/FormsComponent/forms/server	$ORACLE_INSTANCE/config/FormsComponent/forms/server
web.xml	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/formsweb.war/WEB-INF/	$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF
weblogic.xml	$DOMAIN_HOME/servers/WLS_FORMS/stage/formsapp/11.1.1/formsapp/formsweb.war/WEB-INF/	$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF
forms.conf	$ORACLE_INSTANCE/config/OHS/<OHS INSTANCE NAME>/moduleconf	$ORACLE_INSTANCE/config/OHS/<OHS INSTANCE NAME>/moduleconf
jvmcontroller.cfg	$ORACLE_INSTANCE/config/FRComponent/frcommon/tools/jvm/	$ORACLE_INSTANCE/config/FRComponent/frcommon/tools/jvm/
webutil.cfg	$ORACLE_INSTANCE/config/FormsComponent/forms/server/	$ORACLE_INSTANCE/config/FormsComponent/forms/server/
Registry.dat	$ORACLE_INSTANCE/config/FormsComponent/forms/registry/oracle/forms/registry/	$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/forms/registry/oracle/forms/registry

C.2 Default formsweb.cfg

A sample of the default `formsweb.cfg` file contains the following:

```
#formsweb.cfg defines parameter values used by the FormsServlet
# formsweb.cfg defines parameter values used by the FormsServlet (frmservlet)
# This section defines the Default settings.  Any of them may be overridden in the
# following Named Configuration sections.  If they are not overridden, then the
# values here will be used.
# The default settings comprise two types of parameters: System parameters,
# which cannot be overridden in the URL, and User Parameters, which can.
```

```
# Parameters which are not marked as System parameters are User parameters.
# SYSTEM PARAMETERS
# -----------------
# These have fixed names and give information required by the Forms
# Servlet in order to function.  They cannot be specified in the URL query
# string.  But they can be overridden in a named configuration (see below).
# Some parameters specify file names: if the full path is not given,
# they are assumed to be in the same directory as this file.  If a path
# is given, then it should be a physical path, not a URL.
# USER PARAMETERS
# ---------------
# These match variables (e.g. %form%) in the baseHTML file. Their values
# may be overridden by specifying them in the URL query string
# (e.g. "http://myhost.example.com/forms/frmservlet?form=myform&width=700")
# or by overriding them in a specific, named configuration (see below)
[default]
# System parameter: default base HTML file
baseHTML=base.htm
# System parameter: base HTML file for use with Sun's Java Plug-In
baseHTMLjpi=basejpi.htm
# System parameter: delimiter for parameters in the base HTML files
HTMLdelimiter=%
# System parameter: file setting environment variables for the Forms runtime
processes
envFile=default.env

# Forms runtime argument: whether to escape certain special characters
# in values extracted from the URL for other runtime arguments
escapeparams=true
# Forms runtime argument: which form module to run
form=test.fmx
# Forms runtime argument: database connection details
userid=
# Forms runtime argument: whether to run in debug mode
debug=no
# Forms runtime argument: host for debugging
host=
# Forms runtime argument: port for debugging
port=
# Forms runtime argument: BIDI digitSubstitution
digitSubstitution=context
# Other Forms runtime arguments: grouped together as one parameter.
# These settings support running and debugging a form from the Builder:
otherparams=obr=%obr% record=%record% tracegroup=%tracegroup% log=%log%
term=%term% ssoProxyConnect=%ssoProxyConnect%
# Sub argument for otherparams
obr=no
# Sub argument for otherparams
record=
# Sub argument for otherparams
tracegroup=
# Sub argument for otherparams
log=
# Sub argument for otherparams
term=

# HTML page title
pageTitle=Oracle Fusion Middleware Forms Services
# HTML attributes for the BODY tag
HTMLbodyAttrs=
```

```
# HTML to add before the form
HTMLbeforeForm=
# HTML to add after the form
HTMLafterForm=

# Forms applet parameter: URL path to Forms ListenerServlet
serverURL=/forms/lservlet
# Forms applet parameter
codebase=/forms/java
# Forms applet parameter
imageBase=codebase
# Forms applet parameter
width=750
# Forms applet parameter
height=600
# Forms applet parameter
separateFrame=false
# Forms applet parameter
splashScreen=
# Forms applet parameter
allowAlertClipboard=true
# Forms applet parameter
disableValidateClipboard=false
# Forms applet parameter
highContrast=false
# Forms applet parameter
background=
# Forms applet parameter
lookAndFeel=Oracle
# Forms applet parameter
colorScheme=teal
# Forms applet parameter
logo=
# Forms applet parameter
restrictedURLparams=pageTitle,HTMLbodyAttrs,HTMLbeforeForm,HTMLafterForm,log
# Forms applet parameter
formsMessageListener=
# Forms applet parameter
recordFileName=
# Forms applet parameter
serverApp=default
# Forms applet archive setting for other clients (Sun Java Plugin, Appletviewer,
 etc)
archive=frmall.jar
# Number of times client should retry if a network failure occurs.  You should
# only change this after reading the documentation.
networkRetries=0
# Page displayed to users to allow them to download Sun's Java Plugin.
# Sun's Java Plugin is typically used for non-Windows clients.
# (NOTE: you should check this page and possibly change the settings)
jpi_download_page=http://java.sun.com/products/archive/j2se/6u12/index.html
# Parameter related to the version of the Java Plugin
jpi_classid=clsid:CAFEEFAC-0016-0000-0012-ABCDEFFEDCBA
# Parameter related to the version of the Java Plugin
jpi_
codebase=http://java.sun.com/update/1.6.0/jinstall-6-windows-i586.cab#Version=1,6,
0,12
# Parameter related to the version of the Java Plugin
jpi_mimetype=application/x-java-applet;jpi-version=1.6.0_12
# Applet parameter for Sun's Java Plugin
```

```
legacy_lifecycle=false
# Single Sign-On OID configuration parameter: indicates whether we allow
# dynamic resource creation if the resource is not yet created in the OID.
ssoDynamicResourceCreate=true
# Single Sign-On parameter: URL to redirect to if ssoDynamicResourceCreate=false
ssoErrorUrl=
# Single Sign-On parameter: Cancel URL for the dynamic resource creation DAS page.
ssoCancelUrl=
# Single Sign-On parameter: indicates whether the url is protected in which
# case mod_osso will be given control for authentication or continue in
# the FormsServlet if not. It is false by default. Set it to true in an
# application-specific section to enable Single Sign-On for that application.
ssoMode=false
# Single Sign-On parameter: indicates whether session should operate in proxy
# user support or not. Specify ssoProxyConnect=yes to enable for particular
 application.
ssoProxyConnect=no
# The parameter allow_debug determines whether debugging is permitted.
# Administrators should set allow_debug to "true" if servlet
# debugging is required, or to provide access to the Forms Trace Xlate utility.
# Otherwise these activities will not be allowed (for security reasons).
allow_debug=false
# Parameter which determines whether new Forms sessions are allowed.
# This is also read by the Forms EM Overview page to show the
# current Forms status.
allowNewConnections=true
# EndUserMonitoring
# EndUserMonitoringEnabled parameter
# Indicates whether EUM/Chronos integration is enabled
EndUserMonitoringEnabled=false
# EndUserMonitoringURL
# indicates where to record EUM/Chronos data
EndUserMonitoringURL=
# Config for javascript integration
applet_name=
enableJavascriptEvent=true
# Config variable that will indicate if heartbeat will
# be blocked when a javascript call is a blocking call.
# The default value if false, i.e heart beat will not be
# blocked for any javascript calls.
JavaScriptBlocksHeartBeat=false
# Example Named Configuration Section
# Example 1: configuration to run forms in a separate browser window with
# "generic" look and feel (include "config=sepwin" in the URL)
# You may define your own specific, named configurations (sets of parameters)
# by adding special sections as illustrated in the following examples.
# Note that you need only specify the parameters you want to change.  The
# default values (defined above) will be used for all other parameters.
# Use of a specific configuration can be requested by including the text
# "config=<your_config_name>" in the query string of the URL used to run
# a form.  For example, to use the sepwin configuration, your could issue
# a URL like "http://myhost.example.com/forms/frmservlet?config=sepwin".
[sepwin]
separateFrame=True
lookandfeel=Generic
# Example Named Configuration Section
# Example 2: configuration running the Forms ListenerServlet in debug mode
# (debug messages will be written to the servlet engine's log file).
[debug]
serverURL=/forms/lservlet/debug
```

```
# Sample configuration for deployingWebUtil. Note that WebUtil is
# only installed with the Forms Builder and is also available for download
# from OTN.
[webutil]
WebUtilArchive=frmwebutil.jar,jacob.jar
WebUtilLogging=off
WebUtilLoggingDetail=normal
WebUtilErrorMode=Alert
WebUtilDispatchMonitorInterval=5
WebUtilTrustInternal=true
WebUtilMaxTransferSize=16384
baseHTML=webutilbase.htm
baseHTMLjpi=webutiljpi.htm
archive=frmall.jar
lookAndFeel=oracle
```

C.3 Platform Specific default.env Files

There are two platform specific versions of default.env:

- Default default.env File for Windows
- Default default.env File for UNIX and Linux

C.3.1 Default default.env File for Windows

```
# default.env - default Forms environment file, Windows version
#
# This file is used to set the Forms runtime environment parameters.
# If a parameter is not defined here, the value used will be that defined
# in the environment in which the WLS Managed Server was started.
#
# NOTES
# Configuration assistant will replace all the macro's with
#    the actual values.
#
  ORACLE_HOME=D:\Oracle2\Middleware\as_2
  ORACLE_INSTANCE=D:\Oracle2\Middleware\asinst_2

  #
  # TNS Entry to locate the database
  #
  TNS_ADMIN=D:\Oracle2\Middleware\asinst_2\config

  #
  # Search path for Forms applications (.fmx files, PL/SQL libraries)
  # If you need to include more than one directory, they should be semi-colon
  # separated (e.g. c:\test\dir1;c:\test\dir2)
  #
  FORMS_PATH=D:\Oracle2\Middleware\as_2\forms;D:\Oracle2\Middleware\asinst
_2\FormsComponent\forms

  # webutil config file path
  WEBUTIL_CONFIG=D:\Oracle2\Middleware\asinst
_2\config\FormsComponent\forms\server\webutil.cfg

  # Disable/remove this variable if end-users need access to the query-where
  # functionality which potentially allows them to enter arbitrary SQL
  # statements when in enter-query mode.
```

```
  FORMS_RESTRICT_ENTER_QUERY=TRUE

  #
  # The PATH setting is required in order to pick up the JVM (jvm.dll and
  # java.exe). Since PATH is being set, it needs to also include
  # D:\Oracle2\Middleware\as_2\bin so relevant files are correctly found.
  #
  PATH=D:\Oracle2\Middleware\as_2\bin;D:\Oracle2\Middleware\as
2\jdk\jre\bin\client;D:\Oracle2\Middleware\as_2\jdk\bin

  #
  # Settings for Forms tracing and logging
  # ----------------------------------------------
  # Note: By default tracing and logging directory is
  # %ORACLE_INSTANCE%\FormsComponent\forms\trace
  # To change the trace directory this entry has to be uncommented and set to
  # desired directory for tracing and logging

    #FORMS_TRACE_DIR=%ORACLE_INSTANCE%\FormsComponent\forms\trace

  #
  # Settings for Javascript events
  # ----------------------------------------------
  # Note: If this variable is set to false then the triggers and
  # built-ins associated with javascript events are disabled

    #FORMS_ALLOW_JAVASCRIPT_EVENTS=

  #
  # System settings
  # ---------------
  # You should not normally need to modify these settings
  #
  FORMS=D:\Oracle2\Middleware\as_2\forms

  #
  # Java class path
  #   This is required for the Forms debugger
  #   You can append your own Java code here)
  # frmsrv.jar and ldapjclnt11.jar are required for
  # the password expiry feature to work(#2213140).
  #
  CLASSPATH=D:\Oracle2\Middleware\as
_2\forms\j2ee\frmsrv.jar;D:\Oracle2\Middleware\as
_2\jlib\ldapjclnt11.jar;D:\Oracle2\Middleware\as
_2\jlib\debugger.jar;D:\Oracle2\Middleware\as
_2\jlib\ewt3.jar;D:\Oracle2\Middleware\as
_2\jlib\share.jar;D:\Oracle2\Middleware\as
_2\jlib\utj.jar;D:\Oracle2\Middleware\as
_2\jlib\zrclient.jar;D:\Oracle2\Middleware\as
_2\reports\jlib\rwrun.jar;D:\Oracle2\Middleware\as
_2\forms\java\frmwebutil.jar;D:\Oracle2\Middleware\as_2/jlib/start
_dejvm.jar;D:\Oracle2\Middleware\as_2\opmn\lib\optic.jar
```

C.3.2 Default default.env File for UNIX and Linux

```
# default.env - default Forms environment file, Linux version
#
# This file is used to set the Forms runtime environment parameters.
# If a parameter is not defined here, the value used will be that defined
```

```
# in the environment in which the WLS Managed Server was started.
#
# NOTES
#    Configuration assitant will replace all the macro's with
#    the actual values.
#
#
#
  ORACLE_HOME=/as_1
  ORACLE_INSTANCE=/asinst_1

  #
  # TNS Entry to locate the database
  #
  TNS_ADMIN=/asinst_1/config

  #
  # Search path for Forms applications (.fmx files, PL/SQL libraries)
  #
  FORMS_PATH=/as_1/forms:/asinst_1/FormsComponent/forms

  #
  # WebUtil config file path. WebUtil is available for download from OTN.
  #
  WEBUTIL_CONFIG=/asinst_1/config/FormsComponent/forms/server/webutil.cfg

  # Disable/remove this variable if end-users need access to the query-where
  # functionality which potentially allows them to enter arbitrary SQL
  # statements when in enter-query mode.
  FORMS_RESTRICT_ENTER_QUERY=TRUE

  # Java class path
  #   This is required for the Forms debugger
  #   You can append your own Java code here)
  # frmsrv.jar and ldapjclnt11.jar are required for
  # the password expiry feature to work(#2213140).
  #
  CLASSPATH=/as
_1/forms/j2ee/frmsrv.jar:/as
_1/jlib/ldapjclnt11.jar:/as
_1/jlib/debugger.jar:/as
_1/jlib/ewt3.jar:/as_1/jlib/share.jar:/as
_1/jlib/utj.jar:/as
_1/jlib/zrclient.jar:/as
_1/reports/jlib/rwrun.jar:/as
_1/forms/java/frmwebutil.jar:/as_1/jlib/start
_dejvm.jar:/as_1/opmn/lib/optic.jar
#

  # The PATH setting is not required for frmweb if the Forms executables are
  # in <ORACLE_HOME>/bin. JDK/bin is also required for dejvm to be
  # auto-started by frmweb.
  #
  PATH=/scratch/cls0223/bea/as_1/bin:/scratch/cls0223/bea/as_1/jdk/bin

  #
  # Settings for Reports
  # ------------------------------
  # NOTE: This setting is only needed if Reports applications
 # are called from Forms applications
```

```
  # However, because of bug 2336698 where a report is started from
  # a forms debugger session with an already running JVM, then
  # the report's class path should also be included in the forms
  # class path.
  # We no longer need to set REPORTS_CLASSPATH as forms will
  # always start the JVM before calling reports.

  #
  # Settings for Forms tracing and logging
  # ----------------------------------------------
  # Note: By default tracing and logging directory is
  # $ORACLE_INSTANCE/FormsComponent/forms/trace
  # To change the trace directory this entry has to be uncommented and set to
  # desired directory for tracing and logging

    #FORMS_TRACE_DIR=/scratch/cls0223/asinst_1/FormsComponent/forms/trace

  #
  # Settings for Javascript events
  # ----------------------------------------------
  # Note: If this variable is set to false then the triggers and
  # built-ins associated with javascript events are disabled

    #FORMS_ALLOW_JAVASCRIPT_EVENTS=


  #
  # System settings
  # ---------------
  # You should not normally need to modify these settings
  #
    #
    # Path for shared library objects
    #   This is highly platform (if not machine) specific !  At install time
    #   <percent>LD_LIBRARY_PATH<percent> should be replaced with the
    #   actual value of the LD_LIBRARY_PATH environment variable (at install
    #   time).  That should ensure we have the paths for such necessities as
    #   the motif and X11 libraries.
    # Explanations:
    #   - Reports needs the path for libjava.so
    #     (.../jre/lib/sparc)
    #   - Forms needs two paths to the jre, for libjvm.so and libhpi.so
    #   - In JDK 1.4.1 the location of libjvm.so is lib/sparc (there is no
    #     classic directory) so we do not include the .../classic directory
    #     below. There are other versions of libjvm.so (in directories server,
    #     client and hotspot) but we will use the version in lib/sparc for now.
    #
    LD_LIBRARY_PATH=/bea/as_1/lib:/bea/as
_1/jdk/jre/lib/i386:/bea/as
_1/jdk/jre/lib/i386/server:/bea/as_1/jdk/jre/lib/i386/native
_threads
    #
    # Setting to take care of signal-chaining facility offered by JVM 1.5
    # Without this Forms/Reports integration could have issues on Unix/Linux
    #
    LD_PRELOAD=/as_1/jdk/jre/lib/i386/libjsig.so
```

C.4 base.htm and basejpi.htm Files

Two baseHTML files are created for your system by the Oracle Universal Installer during Forms installation and configuration. **In most cases, you will not need to modify these files.** If you do need to modify these files, you should create your own versions and reference them from the `formsweb.cfg` file. The default files may be overridden by a patch installation.

When a user first starts an Oracle Forms application (by clicking a link to the application's URL), a baseHTML file is read by Forms servlet.

Any variables (%*variablename*%) in the baseHTML file are replaced with the appropriate parameter values specified in the `formsweb.cfg` file described in Section 4.2, "Configuring Forms Services", and from query parameters in the URL request (if any). Query parameter values override the values in the `formsweb.cfg` file.

Then, the baseHTML file is downloaded to the user's Web browser.

The following baseHTML starter files are available in the `$ORACLE_INSTANCE/config/FormsComponent/forms/server/` directory:

- **basejpi.htm**: This is the baseHTML file for Java Plug-in. The Forms servlet uses this default file if the client browser is on Windows.
- **base.htm**: This is a baseHTML file containing the APPLET tags required to run the Forms applet in the AppletViewer, or in any Web browser certified by Oracle with a native JVM that is certified with Oracle Forms. See Default base.htm File for an example.

To create a new baseHTML file:

1. Copy the basejpi.htm, or base.htm starter file, which is located in the `$ORACLE_INSTANCE/config/FormsComponent/forms/server/` directory.
2. Rename the file (for example, `order.htm)` .
3. Add or modify any text that is visible to the user (for example, text contained within <TITLE> and <BODY> tags).
4. Modify the parameters as needed. It is recommended that you use variables in the baseHTML file, and specify the actual values in the `formsweb.cfg` file, as described in `formsweb.cfg`.

   The baseHTML tags can also be set in the specific named configuration section, overwriting the system default value. This is recommended if an individual custom baseHTML template needs to be used. However, if a custom template is used for all applications, then it is recommended you change the default configuration section in the `formsweb.cfg` file.
5. Place the new baseHTML file in the `$ORACLE_INSTANCE/config/FormsComponent/forms/server/` directory, update the baseHTML, baseHTMLjpi parameter in the `formsweb.cfg` file to point to the new baseHTML files.

C.4.1 Parameters and variables in the baseHTML file

If you do not want to use a parameter tag that is provided in the base.htm or basejpi.htm file, delete it from the file.

Oracle recommends that you specify the rest of the parameter values as variables (%*variablename*%) in the baseHTML file. For example:

```
<PARAM NAME="logo"  VALUE="%logo%">
```

Then, specify the actual parameter values in the `formsweb.cfg` file. All variables are replaced with the appropriate parameter values at runtime.

C.4.1.1 Usage Notes

- You can use a variable value anywhere in the baseHTML file. Variables are specified as a name enclosed in a special delimiter (the default delimiter is %). For example, you could have the following line in your HTML file:

  ```
  ARCHIVE="%Archive%"
  ```

  You must then assign a value to `%Archive%` either in the formsweb.cfg file or in the URL query string.

- All variables must receive values at runtime. If a variable does not receive a value, Forms Services cannot build a proper HTML file to pass back to the user's Web browser, resulting in an error.
- To streamline performance, use only one Web server as a source for Jar file downloads. This will prevent multiple downloads of the same files from different servers.

C.4.2 Default base.htm File

```
<HTML>
<!-- FILE: base.htm (Oracle Forms) -->
<!--                               -->
<!-- This is the default base HTML file for running a form on
<!--the web using a generic APPLET tag to include   -->
<!-- Forms applet.-->
<!--                              -->
<!-- IMPORTANT NOTES:             -->
<!-- Default values for all the variables which appear
<!-- below (enclosed in percent characters) are defined-->
<!-- in the servlet configuration file (formsweb.cfg).    -->
<!--  It is preferable to make changes in that file where   -->
<!--  possible, rather than this one.     -->

<!-- This file will be REPLACED if you reinstall
<!--Oracle Forms, so you are advised to make your own  -->
<!--  version if you want to make want to make any  -->
<!--  modifications.  You should then set the       -->
<!-- baseHTML parameter in the Forms Servlet
<!--configuration file (formsweb.cfg) to point to     -->
<!--  your new file instead of this one.  -->
<HEAD><TITLE>%pageTitle%</TITLE></HEAD>
<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%
<COMMENT id="forms_plugin_info"
         serverURL="%serverURL%"
         appcodebase="%codebase%"
         apparchive="%archive%"
         appheight="%Height%"
         appwidth="%Width%"
         appname="%applet_name%">
</COMMENT>
<!-- Forms applet definition (start) -->
<NOSCRIPT>
```

```
<APPLET CODEBASE="%codebase%"
        CODE="oracle.forms.engine.Main"
        ARCHIVE="%archive%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        NAME="%applet_name%" MAYSCRIPT>
</NOSCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="/forms/frmjscript/forms_base_ie.js">
</SCRIPT>
<PARAM NAME="serverURL" VALUE="%appletServerURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
VALUE="%escapeParams% module=%form% userid=%userid% debug=%debug% host=%host%
port=%port% %otherParams%">
<PARAM NAME="separateFrame" VALUE="%separateFrame%">
<PARAM NAME="splashScreen"  VALUE="%splashScreen%">
<PARAM NAME="background"  VALUE="%background%">
<PARAM NAME="lookAndFeel"  VALUE="%lookAndFeel%">
<PARAM NAME="colorScheme"  VALUE="%colorScheme%">
<PARAM NAME="serverApp" VALUE="%serverApp%">
<PARAM NAME="logo" VALUE="%logo%">
<PARAM NAME="imageBase" VALUE="%imageBase%">
<PARAM NAME="formsMessageListener" VALUE="%formsMessageListener%">
<PARAM NAME="recordFileName" VALUE="%recordFileName%">
<PARAM NAME="EndUserMonitoringEnabled" VALUE="%EndUserMonitoringEnabled%">
<PARAM NAME="EndUserMonitoringURL" VALUE="%EndUserMonitoringURL%">
<PARAM NAME="heartbeat" VALUE="%heartbeat%">
<PARAM NAME="MaxEventWait" VALUE="%MaxEventWait%">
<PARAM NAME="allowAlertClipboard" VALUE="%allowAlertClipboard%">
<PARAM NAME="disableValidateClipboard" VALUE="%disableValidateClipboard%">
<PARAM NAME="enableJavascriptEvent" VALUE="%enableJavascriptEvent%">
<PARAM NAME="digitSubstitution" VALUE="%digitSubstitution%">
<PARAM NAME="legacy_lifecycle" VALUE="%legacy_lifecycle%">
<PARAM NAME="JavaScriptBlocksHeartBeat" VALUE="%JavaScriptBlocksHeartBeat%">
<PARAM NAME="highContrast" VALUE="%highContrast%">
</Applet>
<!--Forms applet deinition (end) -->
&HTMLafterForm%
</BODY>
</HTML>
```

C.4.3 Default basejpi.htm File

```
<HTML>
<!-- FILE: basejpi.htm (Oracle Forms)   -->
<!--                                    -->
<!-- This is the default base HTML file for running
<!--a form on the  web using the JDK Java Plugin.   -->
<!--  This is used for example when     -->
<!-- running with Netscape on Unix.     -->
<!-- IMPORTANT NOTES:                   -->
<!-- Default values for all the variables which -->
<!--appear below (enclosed in percent characters)-->
<!-- are defined in the servlet configuration file  -->
<!-- (formsweb.cfg). It is preferable to make      -->
<!-- changes in that file where possible, rather than
<!--this one.       -->
<!--                                  -->
<!-- This file will be REPLACED if you reinstall
<!--Oracle Forms, so     -->
```

```
<!-- you are advised to create your own version if
<!--you want to make   -->
<!-- any modifications.  You should then set the
<!--baseHTMLjpi          -->
<!-- parameter in the Forms Servlet configuration file
<!--(formsweb.cfg) -->
<!-- to point to your new file instead of this one. -->
<HEAD>
<TITLE>%pageTitle%</TITLE>
</HEAD>
<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%
<COMMENT id="forms_plugin_info"
         serverURL="%serverURL%"
         plug_ver="%jpi_classid%"
         appheight="%Height%"
         appwidth="%Width%"
         appcodebase="%jpi_codebase%"
         appname="%applet_name%">
</COMMENT>
<!-- Forms applet definition (start) -->
<NOSCRIPT>
<OBJECT classid="%jpi_classid%"
        codebase="%jpi_codebase%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0"
        ID="%applet_name%">
</NOSCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="/forms/frmjscript/forms_ie.js">
</SCRIPT>
<PARAM NAME="TYPE"       VALUE="%jpi_mimetype%">
<PARAM NAME="CODEBASE"   VALUE="%codebase%">
<PARAM NAME="CODE"       VALUE="oracle.forms.engine.Main" >
<PARAM NAME="ARCHIVE"    VALUE="%archive%" >
<PARAM NAME="serverURL" VALUE="%appletServerURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="%escapeParams% module=%form% userid=%userid% debug=%debug%
host=%host% port=%port% %otherParams%">
<PARAM NAME="separateFrame" VALUE="%separateFrame%">
<PARAM NAME="splashScreen"  VALUE="%splashScreen%">
<PARAM NAME="background"  VALUE="%background%">
<PARAM NAME="lookAndFeel"  VALUE="%lookAndFeel%">
<PARAM NAME="colorScheme"  VALUE="%colorScheme%">
<PARAM NAME="serverApp" VALUE="%serverApp%">
<PARAM NAME="logo" VALUE="%logo%">
<PARAM NAME="imageBase" VALUE="%imageBase%">
<PARAM NAME="formsMessageListener" VALUE="%formsMessageListener%">
<PARAM NAME="recordFileName" VALUE="%recordFileName%">
<PARAM NAME="EndUserMonitoringEnabled" VALUE="%EndUserMonitoringEnabled%">
<PARAM NAME="EndUserMonitoringURL" VALUE="%EndUserMonitoringURL%">
<PARAM NAME="heartBeat" VALUE="%heartBeat%">
<PARAM NAME="MaxEventWait" VALUE="%MaxEventWait%">
<PARAM NAME="allowAlertClipboard" VALUE="%allowAlertClipboard%">
<PARAM NAME="disableValidateClipboard" VALUE="%disableValidateClipboard%">
<PARAM NAME="enableJavascriptEvent" VALUE="%enableJavascriptEvent%">
<PARAM NAME="MAYSCRIPT" VALUE="%enableJavascriptEvent%">
<PARAM NAME="digitSubstitution" VALUE="%digitSubstitution%">
```

```
<PARAM NAME="legacy_lifecycle" VALUE="%legacy_lifecycle%">
<PARAM NAME="JavaScriptBlocksHeartBeat" VALUE="%JavaScriptBlocksHeartBeat%">
<PARAM NAME="highContrast" VALUE="%highContrast%">
<COMMENT>
<EMBED SRC="" PLUGINSPAGE="%jpi_download_page%"
        TYPE="%jpi_mimetype%"
        java_codebase="%codebase%"
        java_code="oracle.forms.engine.Main"
        java_archive="%archive%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0
        VSPACE="0"
        NAME="%applet_name%"
serverURL="%appletServerURL%"
        networkRetries="%networkRetries%"
        serverArgs="%escapeParams% module=%form% userid=%userid% debug=%debug%
host=%host% port=%port% %otherparams%"
        separateFrame="%separateFrame%"
        splashScreen="%splashScreen%"
        background="%background%"
        lookAndFeel="%lookAndFeel%"
        colorScheme="%colorScheme%"
        serverApp="%serverApp%"
        logo="%logo%"
        imageBase="%imageBase%"
        recordFileName="%recordFileName%"
        EndUserMonitoringEnabled="%EndUserMonitoringEnabled%"
        EndUserMonitoringURL="%EndUserMonitoringURL%"
        heartBeat="%heartBeat%"
        MaxEventWait="%MaxEventWait%"
        disableValidateClipboard="%disableValidateClipboard%"
allowAlertClipboard="%allowAlertClipboard%"
        enableJavascriptEvent="%enableJavascriptEvent%"
        MAYSCRIPT="%enableJavascriptEvent%"
        digitSubstitution="%digitSubstitution%"
        legacy_lifecycle="%legacy_lifecycle%"
        JavaScriptBlocksHeartBeat="%JavaScriptBlocksHeartBeat%"
        highContrast="%highContrast%"
<NOEMBED>
</COMMENT>
</NOEMBED>
</EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->
%HTMLafterForm%
</BODY>
</HTML>
```

C.5 web.xml

The `web.xml` file is the web application deployment descriptor file for forms Java EE application. This file is located at `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF/`. Advanced users might want to edit the `web.xml` file to:

- Enable extra testing options.

  If you are having difficulty running Oracle Forms in your Oracle Fusion Middleware installation, it can be useful to enable certain test options which are

not usually enabled for security reasons. To use these options, edit the web.xml file to set the testMode frmservlet parameter to true. Then restart the Web server (or Oracle WebLogic Managed Server). The additional options are then visible on the Forms servlet administration page (which can be accessed at a URL like http://<your_web_server_hostname>:<port>/forms/frmservlet/admin).

- Run Oracle Forms using static HTML pages (rather than the Forms servlet).

  When Oracle Forms applications are run using a method other than the Forms servlet (for example, static HTML pages, or JSPs), parameter settings in the `formsweb.cfg` file are not used. You may therefore need to define servlet parameters for the Listener Servlet, such as workingDirectory and envFile (specifying the current working directory for the Forms runtime processes, and the file containing environment settings to be used).

  Servlet mappings are defined in `web.xml`. Table C–2 describes some of the servlet mappings.

Table C–2 web.xml Servlet Mappings

URL Path	Type	Maps to	Purpose
`/forms/frmservlet`	Servlet mount point	Forms servlet	Generate HTML page to run a form
`/forms/lservlet`	Servlet mount point	Forms Listener servlet	Handles message traffic from the Forms applet

C.5.1 Default web.xml File

```
<?xml version='1.0' encoding='UTF-8'?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee">
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <servlet>
    <servlet-name>frmservlet</servlet-name>
    <servlet-class>oracle.forms.servlet.FormsServlet</servlet-class>
    <init-param>
       <!-- Turn on or off sensitive options on the frmservlet/admin page.
            For security reasons this should be set to false for
            production sites.
       -->
       <param-name>testMode</param-name>
       <param-value>false</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>lservlet</servlet-name>
    <servlet-class>oracle.forms.servlet.ListenerServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>frmservlet</servlet-name>
    <url-pattern>/frmservlet/*</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>lservlet</servlet-name>
    <url-pattern>/lservlet/*</url-pattern>
  </servlet-mapping>
  <!-- add mime mapping for the java scripts -->
  <mime-mapping>
```

```
     <extension>js</extension>
      <mime-type>application/x-javascript</mime-type>
  </mime-mapping>
  <welcome-file-list>
    <welcome-file>lservlet</welcome-file>
    <welcome-file>frmservlet</welcome-file>
  </welcome-file-list>
  <listener>

<listener-class>oracle.forms.config.mbeans.FormsappLifeCycleCallBack</listener-cla
ss>
  </listener>
   <!--  Define security constraints to limit access to the defined url to a
   particular role. Logical roles are defined in web.xml and these roles are
mapped
   to actual roles(principal roles) in weblogic.xml
   -->
   <security-constraint>
      <web-resource-collection>
         <web-resource-name>TraceLog</web-resource-name>
         <url-pattern>/frmservlet/trace/*</url-pattern>
         <http-method>GET</http-method>
      </web-resource-collection>
      <auth-constraint>
         <description>Admin users only</description>
         <role-name>formsadmin</role-name>
      </auth-constraint>
   </security-constraint>

   <login-config>
     <auth-method>BASIC</auth-method>
     <realm-name>WebApp</realm-name>
   </login-config>
   <security-role>
        <description>admin role</description>
        <role-name>formsadmin</role-name>
    </security-role>

</web-app>
```

C.6 weblogic.xml

The `weblogic.xml` is the web application deployment descriptor file. This file is located at `$DOMAIN_HOME/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/<random_string>/war/WEB-INF`.

```
<?xml version='1.0' encoding='UTF-8'?>
<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<session-descriptor>
   <timeout-secs>7200</timeout-secs>
   <invalidation-interval-secs>120</invalidation-interval-secs>
   <debug-enabled>false</debug-enabled>
   <id-length>52</id-length>
   <tracking-enabled>true</tracking-enabled>
   <cache-size>1024</cache-size>
   <max-in-memory-sessions>-1</max-in-memory-sessions>
   <cookies-enabled>false</cookies-enabled>
</session-descriptor>
```

```
<!--logical roles defined in web.xml are mapped to the real users below -->
<security-role-assignment>
   <role-name>formsadmin</role-name>
   <principal-name>Administrators</principal-name>
</security-role-assignment>
</weblogic-web-app>
```

C.7 forms.conf

Prior to 11*g*, virtual path mappings were defined in forms.conf. In 11*g*, `forms.conf` defines WebLogic handler mappings for the Managed Server where the Forms Services applications are deployed. For more information, see the Section 3.2.3, "Oracle HTTP Listener Configuration File." The location of the file is `$ORACLE_INSTANCE/config/OHS/<OHS INSTANCE NAME>/moduleconf`.

Note: When including any user-defined `aliasMatch` with the prefix `/forms/` in `forms.conf`, add the directive `WLExcludePathOrMimeType`. For example, in Linux, when defining the `aliasMatch` for `/forms/usericons` in `forms.conf`, the directive `WLExcludePathOrMimeType` is defined as following:

```
AliasMatch /forms/usericons/(..*) "/home/userx/myicons/$1"
WLExcludePathOrMimeType /forms/usericons/
```

C.7.1 Default forms.conf

```
# Name
#    forms.conf  - Forms component Apache directives configuration file.

# Purpose
#    It should include the weblogic managed server (routing) directives for
#    the servers where Forms applications are deployed and other miscellaneous
#    Forms component OHS directives.
#
#
# Remarks
#    This file is included with the OHS configuration under
#    $OI/config/OHS/<OHS Node Name>/moduleconf sub-directory.
#
#
<IfModule !mod_osso.c>
        LoadModule osso_module  ${ORACLE_HOME}/ohs/modules/mod_osso.so
</IfModule>
<IfModule mod_osso.c>
        OssoHTTPOnly off
</IfModule>
<Location /forms>
        SetHandler weblogic-handler
        WebLogicCluster    dadvma0190.us.oracle.com:9001
        DynamicServerList OFF
</Location>
#
# virtual mapping for the /forms/html mapping.
#
RewriteEngine on
```

```
RewriteRule ^/forms/html/(..*) /workaroundhtml/$1 [PT]
AliasMatch ^/workaroundhtml/(..*)
"/scratch/fmw/ps1/rc3/asinst_2/config/FormsComponent/forms/html/$1"
```

C.8 Registry.dat

Location: `$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/forms/registry/oracle/forms/registry`

This file enables you to change the default font, font mappings, and icons that Forms Services uses.

C.8.1 Registry.dat

```
# This is the Registry file.
#
# This file contains the logical [Java] Class name and an associated
# [numerical] identifier that will be used to refer to objects of the
# class in order to reduce the amount of information that needs to be
# repeatedly transmitted to the client.
#
# This file is of the Form understood by java.util.Properties (for now)
#
# The System Level sound file is relative to the CODEBASE
#
# The oracle classes which used to be defined here have now been moved to
# within the code.
#
# #
# Defaults for the Font details, all names are Java Font names.  Each of
# these parameters represents the default property to use when none is
# specified.
# defaultFontname represents the default Java fontName.
# defaultSize     represents the default fontSize.  Note that the size is
#                 multiplied by 100 (e.g. a 10pt font has a size of 1000).
# defaultStyle    represents the default fontStyle, PLAIN or ITALIC.
# defaultWeight   represents the default fontWeight, PLAIN or BOLD.
#
default.fontMap.defaultFontname=Dialog
default.fontMap.defaultSize=900
default.fontMap.defaultStyle=PLAIN
default.fontMap.defaultWeight=PLAIN
#
#
# Default Font Face mapping.
#
# appFontname  represents a comma delimited list of Application Font Names.
# javaFontname represents a comma delimited list of Java Font Names.
#
# The number of entries in the appFontname list should match the number in
# the javaFontname list.  The elements of the list are comma separated and
# *all* characters are taken literally, leading and trailing spaces are
# stripped from Face names.
#
# Note that this file uses the Java 1.1 Font names in order to be able to
# handle the NLS Plane (BUG #431051)
#
default.fontMap.appFontnames=Courier
```

```
New,Courier,courier,System,Terminal,Fixed,Fixedsys,Times,Times New Roman,MS Sans
Serif,Arial
default.fontMap.javaFontnames=MonoSpaced,MonoSpaced,MonoSpaced,Dialog,MonoSpaced,D
ialog,Dialog,Serif,Serif,Dialog,SansSerif
# The Application Level icon files are relative to the DOCUMENTBASE
#   example: icons/
# or an absolute URL.
#   example: http://www.example.net/~luser/d2k_project/
#
default.icons.iconpath=
default.icons.iconextension=gif


#
# Application level settings to control UI features
#
app.ui.lovButtons=false
app.ui.requiredFieldVA=false
#    The background color is specified as an RGB triple.
app.ui.requiredFieldVABGColor=255,0,0
```

C.9 Default jvmcontroller.cfg

A Forms application can be configured to use a specific JVM controller using the `jvmcontroller` parameter. This parameter is specified in `formsweb.cfg`. The parameters that are used by the JVM controller are specified in the JVM controller's configuration file, `jvmcontrollers.cfg`. This file is located at `$ORACLE_INSTANCE/config/FRComponent/frcommon/tools/jvm/`.

```
# jvmcontrollers.cfg defines parameter values used by the JVM Controller(dejvm)

# Default JVM Controller
# This section defines the default settings. Any of them may be overridden
# in the following Named JVM Controller sections.  If they are not overridden,
# then the values here will be used.
[default]


# Example: Named JVM Controller
# This section shows example values for a jvm controller. These
# values overrides any values defined for the default controller.
[example]
jvmoptions=-Xms512m -Xmx1024m

# Classpath settings given here is an example only. This should be
# modified to include the required jar files and should be set in
# platform specific manner.
classpath=/myapps/common/jars/common.jar:/myapps/anapp/jars/anapp.jar
maxsessions=50
logdir=/myapps/anapp/log
logging=off
```

C.10 Default webutil.cfg

The `webutil.cfg` file is one of the files used to configure WebUtil at run time. For more information on the file, see Section 3.2.6, "WebUtil Configuration Files." For information about using WebUtil at design time, see the Oracle Forms Developer Help. This file is located at `$ORACLE_INSTANCE/config/FormsComponent/forms/server/`.

```
# ------------------------------------------------------------------------
# webutil.cfg - WebUtil default configuration file
# ------------------------------------------------------------------------
# This file provides all of the configuration settings for webutil. These are
# divided into the following sections:
# 1. Logging Options
# 2. Installation Options
# 3. File Upload and Download Options
# 1. Server Side Logging Options for logging errors and log messages
# You must set logging.enabled to true to allow mid-tier logging. Without this
# mid-tier logging will not take place no matter what PL/SQL or URL options
# are supplied to switch it on. Once logging is enabled the other settings come
# into play.
#
# Details
# -------
#  logging.file       : Defines the file name and location of the log file.
#                       Note that WebUtil does no log file management. You may
#                       need to manually clean this file up from time to time.
#  logging.enabled    : Can be TRUE or FALSE
#  logging.errorsonly : Can be TRUE or FALSE. Setting to true will ensure that
 #                       only errors and not normal informational log messages
#                       are written to the log file. For product use this would
#                       normally be set to TRUE
#  logging.connections: Can be TRUE or FALSE. Setting to true will cause each
#                       connection from a client using WebUtil to write into
#                       the log as it sets up.
logging.file=
logging.enabled=FALSE
logging.errorsonly=FALSE
logging.connections=FALSE
# 2. Installation Options
# WebUtil needs to download some files to the client in order to perform
# certain integration operations such as OLE or Registry Access. These files
# are downloaded the first time that you access one of the functions that need
# them. You have to define the location of these files on the server and the
# location on the client.
#
# Details
# -------
#  install.syslib.location  : The virtual path to the directory holding the
#                             webutil library files on the server side. This
#                             must either be an absolute URL or a URL that is
#                             relative to the documentbase
#
#  install.syslib.location.client.<os>  :
#                             The path to the directory on the client machine
#                             where webutil library files will be downloaded.
#                             This must either be an absolute path or a path
#                             that is relative to client user profile or HOME.
#                             Directory will be created if necessary along with
#                             other required parent directories.
#                             If the path is not set, it will be treated as
#                             a special case where libraries will be downloaded
#                             to client JRE\bin (windows) or JRE/lib (unix).
#                             If this directory is changed, all the libraries
#                             will be redownloaded again.
#
#  install.syslib.<os>.<package>.<n> :
```

```
#                             The name(s) of the libraries required for
#                             particular webutil beans. The format of this is
#                             name|size|version|showDownloadDialog. Multiple
#                             libraries can be downloaded per package. But
#                            ensure that the <n> values are consecutive and
#                             start at 1
install.syslib.location=/webutil
# Uncomment and change the following if you want to specify a client location
# where the syslib libraries can be downloaded.
#install.syslib.location.client.0=webutil\syslib
#install.syslib.location.client.1=webutil/syslib
# Change size and version if necessary, like when upgrading the library.
# Normally this would not be required since most of these libraries come with
# install itself.
install.syslib.0.7.1=jacob.dll|106496|1.10|true
install.syslib.0.9.1=JNIsharedstubs.dll|65582|1.0|true
install.syslib.0.9.2=d2kwut60.dll|192512|1.0|true
# 3. Upload / Download Options
# You can also add your own libraries in here, e.g.
#install.syslib.0.user.1=testwebutil.dll|204872|1.0|true
# For the file upload and download options you can define the default locations
# on the server that webutil can use as a work area. Optionally you can switch
# upload and download off
# Details
# -------
#  transfer.database.enabled   : Can be TRUE or FALSE - allows you to enable or
#                                disable upload and download from the database
#                                server.
#  transfer.appsrv.enabled     : Can be TRUE or FALSE - allows you to enable or
#                                disable upload and download from the
#                                application server.
#  transfer.appsrv.workAreaRoot: The root of the location in which WebUtil can
#                                store temporary files uploaded from the client.
#                                If no location is specified, application server
#                                user_home/temp will be assumed.
#                                This location is always readable and writable
#                                no matter what the settings in
#                                transfer.appsrv.* are. This setting is
#                                required if you need the Client side
#                                READ/WRITE_IMAGE_FILE procedures.
#  transfer.appsrv.accessControl:Can be TRUE or FALSE - allows you to indicate
#                                that uploads and downloads can only occur from
#                                the directories named in the
#                                transfer.appsrv.read.n and
#                                transfer.appsrv.write.n entries and their
#                                subdirectories. If this setting is FALSE,
#                                transfers can happen anywhere.
#  transfer.appsrv.read.<n>:     List of directory names that downloads can read
#                                from.
#  transfer.appsrv.write.<n>:    List of directory names that uploads can write
#                                to.
#NOTE: By default the file transfer is disabled as a security measure
transfer.database.enabled=FALSE
transfer.appsrv.enabled=FALSE
transfer.appsrv.workAreaRoot=
transfer.appsrv.accessControl=TRUE
#List transfer.appsrv.read.<n> directories
transfer.appsrv.read.1=c:\temp
#List transfer.appsrv.write.<n> directories
transfer.appsrv.write.1=c:\temp
```

```
# 4. Others
# Details
# -------
# BlockAllowHeartBeat     : To continue the heart beat communication with the
#                           server when set to TRUE. By default the value is
#                           set to False. When False there would not be heart
#                           beat communication in blocking mode.
BlockAllowHeartBeat=False
```

C.11 Default webutilbase.htm

This file is located at `$ORACLE_INSTANCE/config/FormsComponent/forms/server/`

This template `.htm` file is used in the WebUtil application section.

```
<HTML>
<!-- FILE: webutilbase.htm (Oracle Forms)                          -->
<!--                                                               -->
<!-- This is the default base HTML file for running a form on the   -->
<!-- web using a generic APPLET tag to include Forms applet.       -->
<!-- and a certificate regsitration applet for the WebUtil utility  -->
<!--                                                               -->
<!-- IMPORTANT NOTES:                                              -->
<!-- Default values for all the variables which appear below       -->
<!-- (enclosed in percent characters) are defined in the servlet    -->
<!-- configuration file (formsweb.cfg). It is preferable to make    -->
<!-- changes in that file where possible, rather than this one.     -->
<!--                                                               -->
<!-- This file uses several extra tags that are not present in the  -->
<!-- default template files.  You should ensure that these are      -->
<!-- present in the configuration that uses this template          -->
<!-- The extra substitution Tags are:                              -->
<!--  %webUtilArchive% = jar file containing the WebUtil code      -->
<!--                     (by default this should be frmwebutil.jar) -->
<!--  %WebUtilLogging% = Defines the current logging mode.         -->
<!--                     Valid values: off|on|console|server|all    -->
<!--                     (on == console)                           -->
<!--  %WebUtilLoggingDetail% = Specifies the level of error logging.-->
<!--                     Valid values: normal|detailed             -->
<!--  %WebUtilErrorMode% = Should errors be displayed in an alert   -->
<!--                     as well as the programmer defined         -->
<!--                     locations                                 -->
<!--                     Valid values: console|server|alert|all     -->
<!--  %WebUtilDispatchMonitorInterval% = Counts in second to        -->
<!--                     indicate how often the monitor thread     -->
<!--                     checks to see if the Forms session is still-->
<!--                     alive. Used with the WebUtil_Session      -->
<!--                     package.                                  -->
<!--  %WebUtilTrustInternal% = Should intranet without domain suffix-->
<!--                     be trusted.                               -->
<!--                     Valid values: true|yes|false|no           -->
<!--  %WebUtilMaxTransferSize% = Size in bytes of file transfer     -->
<!--                     segments. Default and maximum allowed is   -->
<!--                     16384, i.e. 16K.                          -->

<HEAD><TITLE>%pageTitle% - WebUtil</TITLE></HEAD>

<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%
```

```
<!-- Registration applet definition (start) -->
<APPLET CODEBASE="%codebase%"
        CODE="oracle.forms.webutil.common.RegisterWebUtil"
        ARCHIVE="%webUtilArchive%"
        WIDTH="0"
        HEIGHT="0"
        HSPACE="0"
        VSPACE="0">

</APPLET>
<!-- Registration applet definition (end) -->

<COMMENT id="forms_plugin_info"
         serverURL="%serverURL%"
         appcodebase="%codebase%"
         apparchive="%archive%,%webUtilArchive%"
         appheight="%Height%"
         appwidth="%Width%"
         appname="%applet_name%">
</COMMENT>

<!-- Forms applet definition (start) -->
<NOSCRIPT>
<APPLET CODEBASE="%codebase%"
        CODE="oracle.forms.engine.Main"
        ARCHIVE="%archive%,%webUtilArchive%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        NAME="%applet_name%" MAYSCRIPT>
</NOSCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="/forms/frmjscript/forms_base_ie.js"></SCRIPT>
<PARAM NAME="serverURL" VALUE="%appletServerURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="%escapeParams% module=%form% userid=%userid% debug=%debug%
host=%host% port=%port% %otherParams%">
<PARAM NAME="separateFrame" VALUE="%separateFrame%">
<PARAM NAME="splashScreen"  VALUE="%splashScreen%">
<PARAM NAME="background"  VALUE="%background%">
<PARAM NAME="lookAndFeel"  VALUE="%lookAndFeel%">
<PARAM NAME="colorScheme"  VALUE="%colorScheme%">
<PARAM NAME="serverApp" VALUE="%serverApp%">
<PARAM NAME="logo" VALUE="%logo%">
<PARAM NAME="imageBase" VALUE="%imageBase%">
<PARAM NAME="formsMessageListener" VALUE="%formsMessageListener%">
<PARAM NAME="recordFileName" VALUE="%recordFileName%">
<PARAM NAME="EndUserMonitoringEnabled" VALUE="%EndUserMonitoringEnabled%">
<PARAM NAME="EndUserMonitoringURL" VALUE="%EndUserMonitoringURL%">
<PARAM NAME="heartbeat" VALUE="%heartbeat%">
<PARAM NAME="heartBeat" VALUE="%heartBeat%">
<PARAM NAME="MaxEventWait" VALUE="%MaxEventWait%">
<PARAM NAME="allowAlertClipboard" VALUE="%allowAlertClipboard%">
<PARAM NAME="disableValidateClipboard" VALUE="%disableValidateClipboard%">
<PARAM NAME="enableJavascriptEvent" VALUE="%enableJavascriptEvent%">
<PARAM NAME="digitSubstitution" VALUE="%digitSubstitution%">
<PARAM NAME="legacy_lifecycle" VALUE="%legacy_lifecycle%">
<PARAM NAME="JavaScriptBlocksHeartBeat" VALUE="%JavaScriptBlocksHeartBeat%">
<PARAM NAME="highContrast" VALUE="%highContrast%">
<PARAM NAME="disableMDIScrollbars" VALUE="%disableMDIScrollbars%">
```

```
<PARAM NAME="clientDPI" VALUE="%clientDPI%">
<!-- Params specific to webutil -->
<PARAM NAME="WebUtilLogging" VALUE="%WebUtilLogging%">
<PARAM NAME="WebUtilLoggingDetail" VALUE="%WebUtilLoggingDetail%">
<PARAM NAME="WebUtilErrormode" VALUE="%WebUtilErrorMode%">
<PARAM NAME="WebUtilDispatchMonitorInterval"
VALUE="%WebUtilDispatchMonitorInterval%">
<PARAM NAME="WebUtilTrustInternal" VALUE="%WebUtilTrustInternal%">
<PARAM NAME="WebUtilMaxTransferSize" VALUE="%WebUtilMaxTransferSize%">
</APPLET>
<!-- Forms applet definition (end) -->
%HTMLafterForm%
</BODY>
</HTML>
```

C.12 Default webutiljpi.htm

This file is located at `$ORACLE_INSTANCE/config/FormsComponent/forms/server/`

This template `.htm` file is used in the WebUtil application section.

```
<HTML>
<!-- FILE: webutiljpi.htm (Oracle Forms)                              -->
<!--                                                                  -->
<!-- This is the default base HTML file for running a form on the     -->
<!-- web using the JDK Java Plugin. This is used for example when     -->
<!-- running with Netscape on Unix.                                   -->
<!-- and a certificate regsitration applet for the WebUtil utility    -->
<!--                                                                  -->
<!-- IMPORTANT NOTES:                                                 -->
<!-- Default values for all the variables which appear below          -->
<!-- (enclosed in percent characters) are defined in the servlet      -->
<!-- configuration file (formsweb.cfg). It is preferable to make      -->
<!-- changes in that file where possible, rather than this one.       -->
<!--                                                                  -->
<!-- This file uses several extra tags that are not present in the    -->
<!-- default template files.  You should ensure that these are        -->
<!-- present in the configuration that uses this template             -->
<!-- The extra substitution Tags are:                                 -->
<!--  %webUtilArchive% = jar file containing the WebUtil code         -->
<!--                     (by default this should be frmwebutil.jar)   -->
<!--  %WebUtilLogging% = Defines the current logging mode.            -->
<!--                     Valid values: off|on|console|server|all      -->
<!--                     (on == console)                              -->
<!--  %WebUtilLoggingDetail% = Specifies the level of error logging.-->
<!--                     Valid values: normal|detailed                -->
<!--  %WebUtilErrorMode% = Should errors be displayed in an alert     -->
<!--                     as well as the programmer defined            -->
<!--                     locations                                    -->
<!--                     Valid values: console|server|alert|all       -->
<!--  %WebUtilDispatchMonitorInterval% = Counts in second to          -->
<!--                     indicate how often the monitor thread        -->
<!--                     checks to see if the Forms session is still-->
<!--                     alive. Used with the WebUtil_Session         -->
<!--                     package.                                     -->
<!--  %WebUtilTrustInternal% = Should intranet without domain suffix-->
<!--                     be trusted.                                  -->
<!--                     Valid values: true|yes|false|no              -->
<!--  %WebUtilMaxTransferSize% = Size in bytes of file transfer       -->
```

```
<!--                    segments. Default and maximum allowed is   -->
<!--                    16384, i.e. 16K.                           -->

<HEAD><TITLE>%pageTitle% - WebUtil</TITLE></HEAD>

<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%

<!-- Registration applet definition (start) -->
<OBJECT classid="%jpi_classid%"
        codebase="%jpi_codebase%"
        WIDTH="0"
        HEIGHT="0"
        HSPACE="0"
        VSPACE="0">
<PARAM NAME="TYPE"       VALUE="%jpi_mimetype%">
<PARAM NAME="CODEBASE"   VALUE="%codebase%">
<PARAM NAME="CODE"       VALUE="oracle.forms.webutil.common.RegisterWebUtil" >
<PARAM NAME="ARCHIVE"    VALUE="%webUtilArchive%" >
<COMMENT>
<EMBED SRC="" PLUGINSPAGE="%jpi_download_page%"
        TYPE="%jpi_mimetype%"
        java_codebase="%codebase%"
        java_code="oracle.forms.webutil.common.RegisterWebUtil"
        java_archive="%webUtilArchive%"
        WIDTH="1"
        HEIGHT="1"
        HSPACE="0"
        VSPACE="0"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Registration applet definition (end) -->

<COMMENT id="forms_plugin_info"
         serverURL="%serverURL%"
         plug_ver="%jpi_classid%"
         appheight="%Height%"
         appwidth="%Width%"
         appcodebase="%jpi_codebase%"
         appname="%applet_name%">
</COMMENT>

<!-- Forms applet definition (start) -->
<NOSCRIPT>
<OBJECT classid="%jpi_classid%"
        codebase="%jpi_codebase%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0"
        ID="%applet_name%">
</NOSCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="/forms/frmjscript/forms_ie.js"></SCRIPT>
<PARAM NAME="TYPE"       VALUE="%jpi_mimetype%">
<PARAM NAME="CODEBASE"   VALUE="%codebase%">
<PARAM NAME="CODE"       VALUE="oracle.forms.engine.Main" >
<PARAM NAME="ARCHIVE"    VALUE="%archive%,%webUtilArchive%" >
```

```
<PARAM NAME="serverURL" VALUE="%appletServerURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="%escapeParams% module=%form% userid=%userid% debug=%debug%
 host=%host% port=%port% %otherParams%">
<PARAM NAME="separateFrame" VALUE="%separateFrame%">
<PARAM NAME="splashScreen"  VALUE="%splashScreen%">
<PARAM NAME="background"  VALUE="%background%">
<PARAM NAME="lookAndFeel"  VALUE="%lookAndFeel%">
<PARAM NAME="colorScheme"  VALUE="%colorScheme%">
<PARAM NAME="serverApp" VALUE="%serverApp%">
<PARAM NAME="logo" VALUE="%logo%">
<PARAM NAME="imageBase" VALUE="%imageBase%">
<PARAM NAME="formsMessageListener" VALUE="%formsMessageListener%">
<PARAM NAME="recordFileName" VALUE="%recordFileName%">
<PARAM NAME="EndUserMonitoringEnabled" VALUE="%EndUserMonitoringEnabled%">
<PARAM NAME="EndUserMonitoringURL" VALUE="%EndUserMonitoringURL%">
<PARAM NAME="heartBeat" VALUE="%heartBeat%">
<PARAM NAME="MaxEventWait" VALUE="%MaxEventWait%">
<PARAM NAME="allowAlertClipboard" VALUE="%allowAlertClipboard%">
<PARAM NAME="disableValidateClipboard" VALUE="%disableValidateClipboard%">
<PARAM NAME="enableJavascriptEvent" VALUE="%enableJavascriptEvent%">
<PARAM NAME="MAYSCRIPT" VALUE="%enableJavascriptEvent%">
<PARAM NAME="digitSubstitution" VALUE="%digitSubstitution%">
<PARAM NAME="legacy_lifecycle" VALUE="%legacy_lifecycle%">
<PARAM NAME="JavaScriptBlocksHeartBeat" VALUE="%JavaScriptBlocksHeartBeat%">
<PARAM NAME="highContrast" VALUE="%highContrast%">
<PARAM NAME="disableMDIScrollbars" VALUE="%disableMDIScrollbars%">
<PARAM NAME="clientDPI" VALUE="%clientDPI%">
<!-- Params specific to webutil -->
<PARAM NAME="WebUtilLogging" VALUE="%WebUtilLogging%">
<PARAM NAME="WebUtilLoggingDetail" VALUE="%WebUtilLoggingDetail%">
<PARAM NAME="WebUtilErrorMode" VALUE="%WebUtilErrorMode%">
<PARAM NAME="WebUtilDispatchMonitorInterval"
 VALUE="%WebUtilDispatchMonitorInterval%">
<PARAM NAME="WebUtilTrustInternal" VALUE="%WebUtilTrustInternal%">
<PARAM NAME="WebUtilMaxTransferSize" VALUE="%WebUtilMaxTransferSize%">
<COMMENT>
<EMBED SRC="" PLUGINSPAGE="%jpi_download_page%"
        TYPE="%jpi_mimetype%"
       java_codebase="%codebase%"
        java_code="oracle.forms.engine.Main"
       java_archive="%archive%,%webUtilArchive%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0"
        NAME="%applet_name%"
        serverURL="%appletServerURL%"
        networkRetries="%networkRetries%"
        serverArgs="%escapeParams% module=%form% userid=%userid% debug=%debug%
 host=%host% port=%port% %otherparams%"
        separateFrame="%separateFrame%"
        splashScreen="%splashScreen%"
        background="%background%"
        lookAndFeel="%lookAndFeel%"
        colorScheme="%colorScheme%"
        serverApp="%serverApp%"
        logo="%logo%"
```

```
        imageBase="%imageBase%"
        recordFileName="%recordFileName%"
        EndUserMonitoringEnabled="%EndUserMonitoringEnabled%"
        EndUserMonitoringURL="%EndUserMonitoringURL%"
        heartBeat="%heartBeat%"
        MaxEventWait="%MaxEventWait%"
        allowAlertClipboard" VALUE="%allowAlertClipboard%"
        disableValidateClipboard="%disableValidateClipboard%"
        enableJavascriptEvent="%enableJavascriptEvent%"
        MAYSCRIPT="%enableJavascriptEvent%"
        digitSubstitution="%digitSubstitution%"
        legacy_lifecycle="%legacy_lifecycle%"
        JavaScriptBlocksHeartBeat="%JavaScriptBlocksHeartBeat%"
        highContrast="%highContrast%"
        disableMDIScrollbars="%disableMDIScrollbars%"
        clientDPI="%clientDPI%"
        WebUtilLogging="%WebUtilLogging%"
        WebUtilLoggingDetail="%WebUtilLoggingDetail%"
        WebUtilErrormode="%WebUtilErrorMode%"
        WebUtilDispatchMonitorInterval="%WebUtilDispatchMonitorInterval%"
        WebUtilTrustInternal="%WebUtilTrustInternal%"
        WebUtilMaxTransferSize="%WebUtilMaxTransferSize%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->
%HTMLafterForm%
</BODY>
</HTML>
```

Index

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z