Oracle® Application Server Forms Services

Deployment Guide

10*g* Release 2 (10.1.2)

B14032-02

August 2005

ORACLE®

Oracle Application Server Forms Services Deployment Guide 10*g* Release 2 (10.1.2)

B14032-02

Primary Author: Orlando Cordero

Contributor: Suvarna Balachandra, Nishad Desai, Pam Gamer, Art Housinger, Phil Kuhn, Chris Lewis, Hiro Nozaki, Gururaja Padakandla, Ganesh Puram, Slava Podokshik, Grant Ronald, Ananth Satyanarayana, Naseer Syed, Sudarshan Upadhya, Robin Zimmermann

Contents

3 Basics of Deploying Oracle Forms Applications

4 Configuring Forms Services

8 Tracing and Diagnostics

9 Configuring End User Monitoring

10 Performance Tuning Considerations

11 Upgrading to OracleAS Forms Services

A Troubleshooting Oracle Forms Services

B JInitiator

C Sample Configuration Files

Preface

Intended Audience

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

Documentation Accessibility

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

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following manuals:

- *Oracle Application Server Release Notes*
- *Oracle Developer Suite Release Notes*
- Oracle Forms Migrating Forms Applications from Forms6i
- Oracle Forms Developer Online Help, available from the Help menu in Forms Developer.

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 Introduction

This guide is intended to provide information about deploying applications with Oracle Application Server Forms Services. When you choose to deploy applications to the Internet, there are many decisions to be made as to how you will go about it. This guide provides information about those decisions and offers suggestions and methods for configuring your system for Web deployment of your applications.

This chapter contains the following sections:

1.1 The Oracle Internet Platform

With Oracle10*g* Database to manage data, Oracle Developer Suite to build applications, and Oracle Application Server to run them, the Oracle Internet platform is a complete solution for building any type of application and deploying it to the Web. These Oracle tools provide a scalable and highly available infrastructure that enables customers to easily accommodate growing user populations.

Oracle offers a simple, complete, and integrated Internet platform composed of three core products:

1.1.1 Oracle Application Server

Oracle Application Server is a scalable, secure, middle-tier application server. It enables you to deliver Web content, host Web applications, and connect to back-office applications. Forms Services are an integral part of the Oracle Application Server bundle, which provides the technology to fully realize the benefits of Internet computing.

1.1.2 Oracle Developer Suite

Oracle Developer Suite combines the power of Oracle Application Development tools, Oracle Business Intelligence tools, the award-winning Oracle XML Developer's Kit (XDK) and the Oracle Application Server Portal Developer Kit (PDK) in one product.

Oracle Developer Suite is based on Internet standards including J2EE, XML, SOAP, UDDI, and UML, and provides a highly productive environment to build applications for Oracle Application Server and Oracle Database 10*g*.

1.1.3 Oracle Database 10*g*

Oracle Database 10g is the latest generation of the world's most popular RDBMS. Among the numerous new capabilities are unlimited scalability and industry-leading reliability with Oracle10*g* Real Application Clusters; new high availability technology including advancements in standby database technology (Oracle Data Guard); and built-in OLAP, data mining and ETL functions.

Oracle Application Server is the best application server for the Oracle Database 10*g* and applications built with Oracle development tools. By leveraging a common technology stack, Oracle Application Server can transparently scale an Oracle Database by caching data and application logic on the middle tier.

1.2 Oracle Application Server Forms Services

As part of Oracle Application Server, Oracle Application Server Forms Services is a new generation of tools that enable you to deploy new and existing Forms Services applications on the World Wide Web.

Forms Services is a comprehensive application framework optimized to deploy Forms applications in a multi-tiered 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.

1.2.1 What's New in Forms Services

Much of the functionality that was handled by the Web server in Forms 6*i* has been assumed by components that are delivered with Oracle Application Server. For example, load balancing, security, scalability, HTTP/S communication handling, and deployment of Java servlets are all performed by various components delivered with OracleAS, such as the Oracle HTTP Server and Oracle Application Server Containers for J2EE (OC4J).

The Forms Services component of OracleAS handles all processing that is specific to Forms Developer applications, such as running the business logic defined in the Forms Developer application and providing the connection to the Oracle Database. A Java applet provides the client user interface.

New features for Forms Services include:

- Runtime Pooling (see Section 10.1.2, "Forms Services Web Runtime Pooling")
- Improved Enterprise Manager Web interface (see Section 6.4, "Enabling OracleAS Single Sign-On for an Application"
- OracleAS Single Sign-On improvements (see Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On")

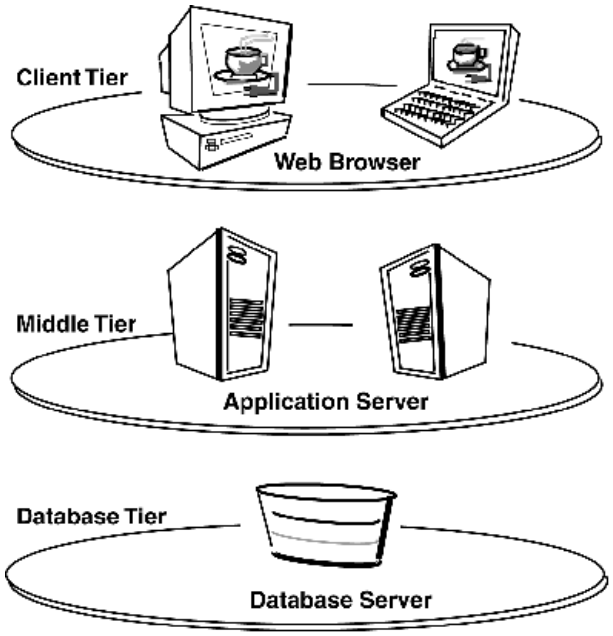
- Improved Integration with Enterprise Manager for easier administration and manageability (see Chapter 8, "Tracing and Diagnostics" and Chapter 9, "Configuring End User Monitoring")
- Tracing and logging improvements (see Chapter 10, "Performance Tuning Considerations")
- Java Virtual Machine (JVM) Pooling (see Chapter 7, "JVM Pooling")

1.3 OracleAS Forms Services Architecture

Forms Services use a three-tier architecture to deploy database applications. Figure 1–1 shows the three tiers that make up the Forms Services architecture:

- The **client tier** contains the Web browser, where the application is displayed.
- The **middle tier** is the application server, where application logic and server software are stored.
- The **database tier** is the database server, where enterprise data is stored.

Figure 1–1 OracleAS Forms Services Architecture



1.4 OracleAS Forms Services Components

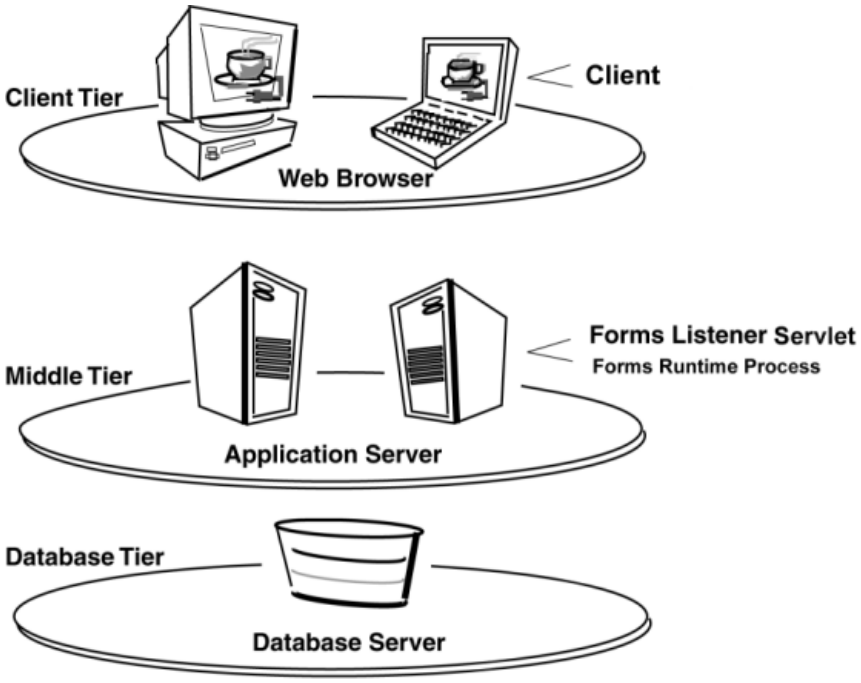
Oracle Application Server Forms Services is a middle-tier application framework for deploying complex, transactional forms applications to the Internet. Developers can build new applications with Forms Developer and deploy them to the Internet with Forms Services. Developers can also take current applications that were previously deployed in client/server and move them to a three-tier architecture without changing the application code.

OracleAS Forms Services consists of three major components, as shown in Figure 1–2:

- The **Client**, which resides on the client tier
- The **Forms Listener Servlet**, which resides on the middle tier

- The **Forms Runtime Process**, which also resides on the middle tier

Figure 1–2 Three-tier configuration for running a form



1.4.1 Forms Listener Servlet

The Forms Listener Servlet acts as 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.

1.4.2 Forms Runtime Process

The Forms runtime process manages application logic and processing. It maintains a connection to the database on behalf of the Java client. It uses the same forms, menus, and library files that were used for running in client/server mode.

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.

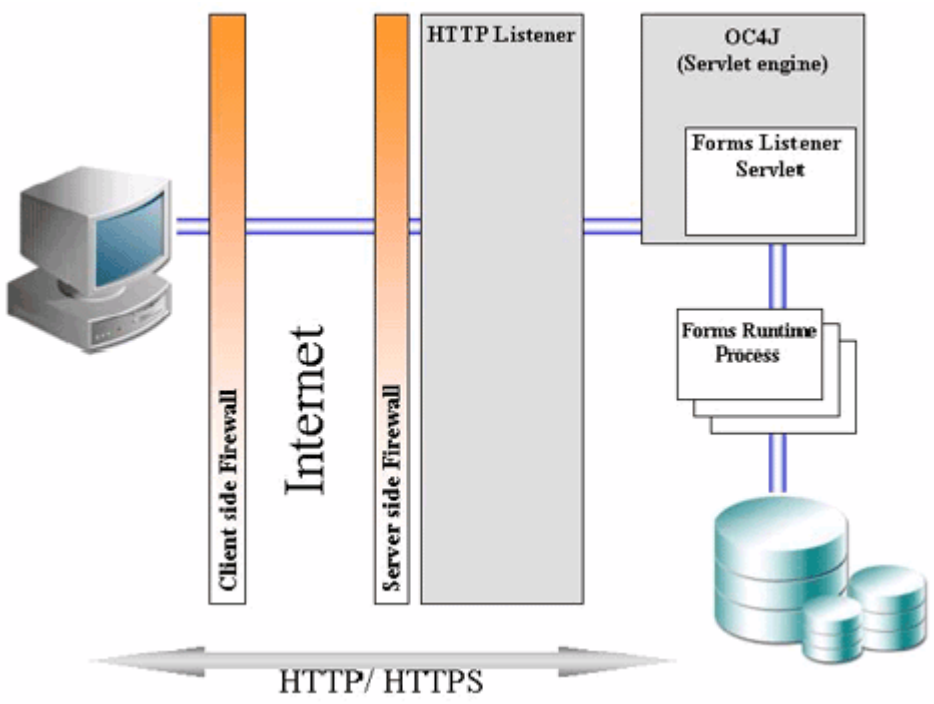
1.5 Forms Listener Servlet

OracleAS Forms Services uses the Forms Listener Servlet (a Java servlet) to start, stop, and communicate with the Forms runtime process. The Forms runtime is what 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. The Forms Listener Servlet replaces the Forms Listener provided in previous releases of Oracle Forms.

Note: You do not need to configure the Forms Listener Servlet as it is already set up for you in the OracleAS installation process.

Figure 1–3 illustrates how the client sends HTTP requests and receives HTTP responses from the Forms Server process. The HTTP Listener acts as the network endpoint for the client, keeping the other server computers and ports from being exposed at the firewall.

Figure 1–3 Architecture using the Forms Listener Servlet



2

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 OracleAS Forms Services:

- Section 2.1, "About OracleAS Forms Services Security"
- Section 2.2, "Configuring OracleAS Forms Services Security"

> **See Also:** For additional information about security, refer to the following documents:
>
> - The *Oracle Application Server Security Guide* provides an overview of Oracle Application Server security and its core functionality.
> - The *Oracle Identity Management Concepts and Deployment Planning Guide* provides guidance for administrators of the Oracle security infrastructure.

2.1 About OracleAS Forms Services Security

This section describes the OracleAS Portal features that you can use to secure your Forms applications when you enable Single Sign-on.

2.1.1 OracleAS Forms Services Single Sign-On

Single Sign-on in Oracle Application Server Forms Services is available through `mod_osso`, an Oracle module for the Oracle HTTP Server. `mod_osso` authenticates a user against Oracle Application Server Single Sign-On, 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 OracleAS Single Sign-On environment, the Forms servlet queries Oracle Internet Directory for the value of the combined unique key that is constructed from the user's OracleAS Single Sign-On 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 OracleAS Single Sign-On environment.

2.1.2 Classes of Users and Their Privileges

Historically, Forms applications use the database to authenticate and authorize application users. To use Oracle Application Server Forms Services with OracleAS Single Sign-On, the user account and its connect information must be available in Oracle Internet Directory. The 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 OracleAS Single Sign-On 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 Application Server Single Sign-On. 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 OracleAS Single Sign-On credentials; however, all users share a common database connect for the application defined by a default RAD entry.

2.1.3 Resources That Are Protected

When you enable OracleAS Single Sign-On for your Forms applications, you can secure your Forms applications with these features:

2.1.3.1 Dynamic Directives

The dynamic `mod_osso` directive runs OracleAS Single Sign-On protected Forms applications as well as non OracleAS Single Sign-On protected Forms applications from the same Oracle Application Server Forms Services instance while using the same configuration files and Forms Servlet. Single sign-on is enabled for applications by a OracleAS Single Sign-On parameter in the application definition of the `forms/server/formsweb.cfg` configuration file.

2.1.3.2 Dynamic Resource Creation in Oracle Internet Directory

In previous releases of Oracle Application Server 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 authorization to do so. In this release of Oracle Application Server Forms Services, you can now configure Oracle Application Server Forms Services to allow users to create the RAD for this application on the fly if it doesn't exist.

2.1.3.3 Database Password Expiration when Using Single Sign-On

In previous releases of Oracle Application Server 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 Application Server 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 Application Server Forms Services.

2.1.4 Authorization and Access Enforcement

For detailed information about the authentication flow of OracleAS Single Sign-On support in Oracle Application Server Forms Services, such as when the first time the user requests an Oracle Application Server Forms Services URL, or from a partner application, see Section 6.6, "Authentication Flow".

2.1.5 Leveraging Oracle Identity Management Infrastructure

Oracle Application Server Forms Services has tighter integration with Oracle Internet Directory with minimal configuration. When you configure OracleAS Single Sign-On for your Forms applications, Oracle Application Server Forms Services handles much of the configuration and interaction with Oracle Internet Directory. For more information about configuring OracleAS Single Sign-On and Oracle Internet Directory, see Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On".

2.2 Configuring OracleAS Forms Services Security

Configuring security for OracleAS Forms Services is done through Oracle Enterprise Manager 10*g* Application Server Control Console. Online help is available for each screen. For more information, see Chapter 4, "Configuring Forms Services" and Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On".

2.2.1 Configuring Oracle Identity Management Options for Oracle Forms

OracleAS 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 6, "Using Forms Services with Oracle Application Server Single Sign-On".

2.2.2 Configuring Oracle Forms Options for OracleAS Security Framework

For more detailed information about configuring and securing Oracle Forms, see the following chapters:

- Chapter 4, "Configuring Forms Services"
- Chapter 5, "Using OracleAS Forms Services with the HTTP Listener and OC4J"
- Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On"
- Chapter 8, "Tracing and Diagnostics"

3

Basics of Deploying Oracle Forms Applications

This chapter describes the basic files you need to configure Oracle Forms, provides an overview of how Forms Services run in Oracle Application Server, and describes the steps you need to follow to deploy Forms applications. After installation is complete, 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 OracleAS Forms Services in Action

This sections describes how Forms Services run in OracleAS, and how the configuration files are used, assuming that the Forms Servlet is used to generate the initial HTML page. For simplicity, we assume the Web server is running on port 7777 on a computer called "mycomputer.com". We also assume no modifications have been made to the standard configuration created during the Oracle Application Server installation process.

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

1. The user starts up their Web browser and goes to a URL like the following:

   `http://mycomputer.com:7777/forms/frmservlet?config=myapp&form=hrapp`
   In this case, 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 forwards the request to OC4J, since the path `/forms/frmservlet` matches one of the OC4J mount directives in the forms.conf file (the one for the Forms Servlet).

3. OC4J maps the request to the Oracle Application Server Forms Services application (whose context root is `/forms`). It maps the request to the Forms Servlet (using the frmservlet mapping specified in the web.xml file).

4. The Forms Servlet (running in OC4J) processes the request as follows:

- Opens the servlet configuration file (formsweb.cfg by default). If that parameter is not set, the default configuration file (`ORACLE_HOME/forms/server/formsweb.cfg`) is used.
- Determines which configuration section to use in the formsweb.cfg file. Since the URL contains the query parameter `config=myapp`, the [myapp] section will be used.
- Determines which baseHTML file to use, based on (a) what browser made the request, (b) what platform the browser is running on, and (c) the settings of various parameters in the formsweb.cfg file (specifically, baseHTMLie, baseHTMLjinitiator, baseHTMLjpi, baseHTML, and IE).
- Reads the baseHTML file, and sends the contents back as an HTML page to the user's Web browser, after doing 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.

For example, the baseHTML file contains the text %form%. In our example, this is replaced with the value "hrapp".

1. Depending on which baseHTML file the Forms Servlet selected, the HTML page sent back to the Web browser will contain an Applet, Object or Embed tag to start up the Forms applet (thin client). The Forms applet runs in a JVM (either the Web browser's native JVM, or a "plugged in" JVM like Oracle JInitiator or Sun's Java plug-in).
2. If the baseHTML file selected was for a plug-in (Oracle JInitiator or Sun's JDK Java plug-in), and if the user does not already have that plug-in installed on their computer, they are prompted to install the plug-in. In the case of JInitiator, the download location is under the virtual path `/forms/jinitiator` (a virtual path defined in the forms.conf file).
3. In order to start up the Forms applet, its Java code must first be loaded. The location of the applet is specified by the applet codebase and archive parameters. For example, if the user is running with Oracle JInitiator, the applet code is loaded from the file http://mycomputer.com:7777/forms/java/frmall_jinit.jar

   The virtual path definition in the formsweb.cfg file for "/forms/java" allows the applet code to be loaded from the Web server.

   Note: The Forms applet code (for example, frmall_jinit.jar) is only to be loaded over the network the first time the user runs an Oracle Application Server Forms Services application (or if a newer version of Oracle Application Server Forms Services is installed on the Web server). Otherwise, it is to be loaded from the Web browser's (or the Java plug-in's) cache on the local disk.
4. Once the Oracle Application Server Forms Services applet is running, it starts up a Forms session by contacting the Forms Listener Servlet at URL `http://mycomputer.com:7777/forms/lservlet`.
5. The Oracle HTTP Server listener receives the request. It forwards the request to OC4J, since the path “/forms/lservlet” matches one of the OC4J mount directives in the forms.conf file (the one for the Forms Listener Servlet).
6. The Forms Listener Servlet (lservlet) starts up a Forms runtime process (frmweb.exe or frmweb) for the Forms session.

7. Communication continues between the Forms applet (running in the user's Web browser) and the Forms runtime process, via the Listener Servlet, until the Forms session ends.
8. The command line (such as giving the name of the form to run) is passed to the Forms runtime process. It is given as the applet parameter "serverArgs". Part of the serverArgs value in the baseHTML file was %form%, which was replaced by "hrapp". Therefore the runtime process actually runs the form in the file "hrapp.fmx".

   This file must either be present in the workingDirectory (which is specified in the Forms Web Configuration page of Application Server Control Console), or in one 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 the Forms Web Configuration page (for example, `form=c:\<path>\myform`).
9. The Forms sessions end when one of the following occurs:
   - The top level form is exited (for example, by PL/SQL trigger code which calls the "exit_form" built-in function). In this case, the user is prompted to save changes if there are unsaved changes. `exit_form(no_validate)` exits the form without prompting.
   - The user quits their Web browser. In this case, any pending updates are lost.

3.2 Configuration Files

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

This section contains the following:

- Section 3.2.1, "Oracle Forms Configuration Files"
- Section 3.2.2, "Oracle Application Server Containers for J2EE (OC4J) Configuration Files"
- Section 3.2.3, "Oracle HTTP Listener Configuration Files"
- Section 3.2.4, "Standard Fonts and Icons File"

Note: Location of files are given relative to the `ORACLE_HOME` directory. Forward slashes should be replaced by back slashes on Windows.

3.2.1 Oracle Forms Configuration Files

Oracle Forms configuration files allow you to specify parameters for your Forms, which you manage through the Application Server Control Console. These configuration files include:

- default.env
- formsweb.cfg
- base.htm, basejini.htm, and basejpi.htm
- ftrace.cfg

Note: If you manually edit any of the configuration or environment files, you'll need to restart Enterprise Manager so that Enterprise Manager can read all changes. If you do not restart Enterprise Manager, any changes that you make through Enterprise Manager will overwrite any manual changes you've made to these files.

3.2.1.1 default.env

Location: `forms/server`

This file contains environment settings for Forms runtime and can be found in the `ORACLE_HOME/forms/server` directory. On Solaris, default.env should include the `PATH` and `LD_LIBRARY_PATH`.

3.2.1.2 formsweb.cfg

Location: `forms/server`.

This is the Forms Servlet configuration file that contains the following:

- Values for Forms runtime command line parameters, as well as 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 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 manage the formsweb.cfg file through Enterprise Manager Application Server Control Console.

For more information about formsweb.cfg, see, Chapter 4.3.1, "Configuring Parameters with Application Server Control Console".

3.2.1.3 base.htm, basejini.htm, and basejpi.htm

Location: `forms/server`.

The baseHTML files (base.htm, basejini.htm, and basejpi.htm) are used as templates by the Forms Servlet when generating the HTML page used to start up an Oracle Forms application.

We recommend that you make configuration changes in the formsweb.cfg file and avoid editing the baseHTML files. If you need to change the baseHTML files, create your own versions and reference them from the formsweb.cfg file by changing the appropriate settings.

For a look at a sample baseHTML files, see Appendix C.3, "base.htm, basejini.htm, and basejpi.htm Files".

3.2.1.4 ftrace.cfg

Location: *forms/server.*

This file allows you to configure Forms Trace. Forms Trace allows you to replace 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, steps the user took while using the form).

You manage Forms Trace through Enterprise Manager Application Server Control Console.

For more information about ftrace.cfg, see Chapter 8, "Tracing and Diagnostics".

3.2.2 Oracle Application Server Containers for J2EE (OC4J) Configuration Files

By default Forms Services is configured for OC4J by deploying it as a J2EE compliant application packaged in an EAR (Enterprise Archive) file called formsapp.ear. This EAR file is deployed during the Oracle Application Server installation process (if you choose to configure Oracle Forms). During deployment, the EAR file is unpacked into the applications directory of the OC4J instance.

This section describes:

- web.xml
- Directory structure for Oracle Forms OC4J files

3.2.2.1 web.xml

Location: `j2ee/OC4J_BI_FORMS/applications/formsapp/formsweb/WEB-INF/web.xml`.

Once Forms Services has been installed and configured, the web.xml file is located in the directory `j2ee/OC4J_BI_FORMS/applications/formsapp/formsweb/WEB-INF` in `ORACLE_HOME`. It defines the aliases `frmservlet` and `lservlet` for the Forms Servlet and the Forms Listener Servlet.

For more information about web.xml, see Appendix C.4, "web.xml".

3.2.2.2 Directory structure for Oracle Forms OC4J files

During Oracle Application Server installation and configuration, the Forms EAR file (formsapp.ear) is deployed to the `"OC4J_BI_FORMS"` OC4J instance. This results in the following directory structure.

Names with a + sign are directories:

```
ORACLE_HOME/j2ee/OC4J_BI_FORMS/applications/formsapp
+META-INF
-application.xml (defines the structure of the ear file)
+formsweb
+WEB-INF
-web.xml (forms & listener servlet definitions, including servlet parameters)
-orion-web.xml (virtual directory mappings and context parameter, only used in
iDS)
+lib
       -frmsrv.jar (contains the Forms Servlet and Listener Servlet code)
```

3.2.3 Oracle HTTP Listener Configuration Files

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

3.2.3.1 forms.conf

Location: `forms/server`.

This is the Oracle HTTP listener configuration file for Oracle Application Server Forms Services. It is included into oracle_apache.conf, which in turn is included into httpd.conf (the master HTTP listener configuration file). forms.conf defines virtual directories (aliases) and servlet mount points to map URL requests to the Forms Servlets running in the OC4J servlet engine.

For more information about forms.conf, see Appendix C.5, "forms.conf".

3.2.4 Standard Fonts and Icons File

This section describes the file used to configure font and icon settings for Oracle Application Server Forms Services.

3.2.4.1 Registry.dat

Location: `forms/java/oracle/forms/registry`

This file allows you to change the default font, font mappings, and icons that Forms Services uses.

For more information about Registry.dat, see Appendix C.6, "Registry.dat".

3.2.5 WebUtil Configuration Files

This section describes the files used to configure WebUtil at runtime. For information about using WebUtil at design time, see the Oracle Forms Online Help.

These files are only available in Oracle Developer Suite under `Oracle_Home/forms/server` directory. These files are *not* available with Oracle Application Server or with Oracle Application Server Forms and Reports Services.

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

3.2.5.1 Default webutil.cfg

Location: `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 more information, see Appendix C.8, "Default webutil.cfg".

3.2.5.2 Default webutilbase.htm

Location: `forms/server`.

This is the default base HTML 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 more information, see Appendix C.9, "Default webutilbase.htm".

3.2.5.3 Default webutiljini.htm

Location: `forms/server`.

This is the HTML template file for running a form on the Web using JInitiator-style tags to include the Forms applet with a certificate registration for WebUtil.

For more information, see Appendix C.10, "Default webutiljini.htm".

3.2.5.4 Default webutiljpi.htm

Location: `forms/server`.

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 a form on the Web with Netscape on Solaris and a certificate registration for WebUtil.

For more information, see Appendix C.11, "Default webutiljpi.htm".

3.3 Application Deployment

Once you have created your application in Forms Developer, you are ready for application Web deployment. Oracle Application Server Forms Services accesses an application in Oracle Application Server through a specified URL. The URL then accesses the HTTP Listener, which communicates with the Listener Servlet. The Listener Servlet starts up a new Forms runtime process (frmweb.exe on Windows or frmweb.sh on Solaris) for each new Forms Services session.

For more information about how Forms Services run, see Section 3.1, "OracleAS 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.

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

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

2. Create a configuration section in the Forms Web Configuration page of Oracle Enterprise Manager 10*g* Application Server Control Console so that Oracle Application Server Forms Services can access your application module.

   Table 3–1 shows you what you would need to configure for an application called "application" with a form module called "form=hrapp.fmx":

Table 3–1 Example New Configuration Section Parameter Values

Configuration Section Name	Application Name	Forms Module Name Value
my_application	application	hrapp.fmx

When configured, the Oracle Application Server Forms Services module hrapp.fmx will be accessible on the Web by entering "...?config=my_application" in the Browser URL (the name of the Forms Web Configuration section in formsweb.cfg).

Note: You can name the configuration section anything you want, as long as it does not include spaces.

3. Make sure the .fmx file location is specified in the FORMS_PATH environment variable. For example, if your .fmx file is located in `d:\my_files\applications`, in the `FORMS_PATH` you would include `d:\my_files\applications` (separated by semi-colons if listing more than one location). You specify this information in the **Forms Edit Environment File** page for that environment file.
4. To modify an environment file, select it in the **Environment** page of Enterprise Manager and add or edit environment variables as needed by your application. For example, you'd add the following environment variables for the previous example, as shown in Table 3–2:

Table 3–2 Example New Environment Variable Values

Environment Variable Name	Environment Variable Value
form	hrapp.fmx

If you specified these environment variables in a new environment file, you will need to specify this environment file in the respective Forms Web configuration section.

5. Enter the name of your application into the following URL:

   `http://mycomputer.com:7777/forms/frmservlet?`

   where "mycomputer" is the name of your computer and "7777" is the port used by your HTTP Listener.

   Once you've created a configuration section, you will need to add "config=" and the name of the configuration section. So, using the example in step 2, the URL to access hrapp.fmx would be:

   `http://mycomputer.com:7777/forms/frmservlet?config=application`

3.3.2 Specifying Parameters

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

- Editing your application settings in the default section of the Forms Web Configuration page of Enterprise Manager Application Server Control Console.

  The default configuration section displays the default values that will be used by Oracle Application Server Forms Services.

  For example, the default value of the system parameter that specifies how to execute the Forms applet under Microsoft Internet Explorer 5.x or above is defined as follows:

  `IE=JInitiator`
  If you want the Forms applet to run in the browser's native JVM, edit the parameter in the IE Value column to read:

  `native`

and click **Apply**.

- You can manage (add, edit, delete) other system and user parameter values in the named application configuration section (see Section 3.3.3, "Creating Configuration Sections in Enterprise Manager"). 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
baseHTMLjinitiator	mybasejini.htm
baseHTMLjpi	mybasejpi.htm
form	myapp.fmx
userid	

> **Note:** Parameters specified in the named configuration section of a Forms Web configuration will override the system parameter settings.

- Override system parameter settings if your application requires modifications to the underlying HTML templates or another value set for the Internet Explorer virtual machine. Change the system parameter setting only if the modification must be adopted by all applications run by the server.

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

3.3.3 Creating Configuration Sections in Enterprise Manager

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

For example, you can make the look and feel of the application to be 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 you can edit, see Chapter 4.3.4, "Default Forms Configuration Parameters".

> **Note:** Parameters specified in the configuration section will override your application default settings.

3.3.3.1 Editing the URL to Access Oracle Application Server Forms Services Applications

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

```
http://mycomputer.com:7777/forms/frmservlet?config=hr&pageTitle="My Company"
```

You can use the ampersand (&) to call a combination of a form and named configuration parameters. For example, in the following URL:

```
http://mycomputer.com:7777/forms/frmservlet?config=myapp&form=hrapp
```

you are calling the form "hrapp" with the parameter settings you specified in "myapp".

Note: Parameters specified in the URL will override parameters set in the configuration section. See Chapter 4.6, "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.

Note: 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 (e.g. base.htm).
2. In the value of a variable specified in the configuration file (e.g. formsweb.cfg), which is substituted (directly or recursively) for a variable reference in (1). Such values are typically maintained using Application Server Control Console; see Chapter 4.3, "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 hex 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 Application Server 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 wish 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 of 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 (' ') and quote ("") must be specified as a URL escape sequence (%20 and %22, respectively). The HTML delimiter character (specified in the configuration file) must also be specified as a URL escape sequence.

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.

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 hex digits that represent a 7-bit ASCII value. Put another way, variable names must never begin with two hex digits, the first of which is in the range 0-7. Put still another way, variable names must never begin with an octal digit followed by a hex digit.

If the HTML delimiter is not %, then an occurrence of % must be escaped if it's immediately followed by an octal digit and then a hex 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

Prior releases 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 3 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 counting 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 Enterprise Manager).

If you wish 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 will tolerate) the new behavior 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 (or modifying existing ones, such as base.htm), 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.

It is permissible to omit the reference to the `escapeparams` variable from the beginning of the value of the `serverArgs` applet parameter, but then you will always obtain 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 wish to obtain the new behavior, 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

Character 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">
```

3.4 Client Browser Support

Users can view Oracle Forms applications on the Web using Oracle JInitiator plug-in (using Netscape Navigator or Internet Explorer). In future patch releases other virtual machines will 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...**

3.4.1 Oracle JInitiator

Oracle JInitiator runs within a Web browser and is based on Sun's JDK/JRE 1.3. It provides the ability to specify a specific Java Virtual Machine (JVM) on the client rather than using the browser's (native) default JVM. Oracle JInitiator does not replace or modify the default JVM provided by the browser. Rather, it provides an alternative JVM in the form of a plug-in for Netscape Navigator and as an ActiveX component for Internet Explorer.

Oracle provides two JAR files (frmall.jar and frmall_jinit.jar) that group and zip classes together for efficient delivery across the network to the client. frmall_jinit.jar is an extra-compressed JAR file that can be used only with Oracle JInitiator to provide increased performance at download time. Once on the client, the files are cached for future use.

For more information about Oracle JInitiator, see Appendix B, "JInitiator".

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

When an 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. Reads the formsweb.cfg file to determine the Internet Explorer parameter setting if the user is using Internet Explorer 5.5 or higher;
3. Selects the appropriate baseHTML file using Table 3–5:

Table 3–5 Web Browsers and the Appropriate baseHTML Files for Each

Browser detected	IE parameter setting	Base HTML file used
Internet Explorer 5.x or 6*	jinitiator	basejini.htm
Netscape Navigator or Internet Explorer version preceding version 5	not applicable	basejini.htm
All other browsers	not applicable	base.htm

* Internet Explorer 6 that has been upgraded from 5.5 *only* (IE 6 is not certified in the base release)

** Internet Explorer running on Windows with the Microsoft Native VM.

1. 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).
2. Sends the HTML file to the user's browser.

4

Configuring Forms Services

This chapter contains the following sections:

4.1 How Oracle Application Server 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 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 10*g* Application Server Control Console as your needs change.

4.2 Enterprise Manager and Oracle Forms

The Enterprise Manager Application Server Control Console user interface that is shipped with Forms Services is a Web-based tool that you launch from your default browser. The default URL is:

```
http://<computer.domain>:1156
```

Note: For information on how to launch Enterprise Manager, see the *Oracle Enterprise Manager Advanced Configuration*.

For Forms Services, use the Web-based Enterprise Manager Application Server Control Console to:

- Monitor metrics for a Forms Services instance. See Chapter 10.1.1.1, "Monitoring Forms Services Instances" for more information.
- Monitor metrics for user sessions. See Chapter 10.1.1.3, "Monitoring Metrics for User Sessions" for more information.
- Allow or deny new user sessions. See Chapter 4.5.1, "Allowing New Users Sessions" and Chapter 4.5.2, "Disabling New User Sessions" for more information.
- Terminate user sessions. See Chapter 4.5.3, "Terminating a User Session on a Forms Services Instance" for more information.
- Configure parameters for a Forms Services instance. See Chapter 4.3.1, "Configuring Parameters with Application Server Control Console" for more information.
- Configure Forms Trace and monitor trace metrics. See Chapter 8.2, "Configuring Forms Trace" and Chapter 8.6, "Monitoring Forms Services Trace Metrics" for more information.
- Configure multiple environment files. See Chapter 4.4, "Configuring Environment Variables with Enterprise Manager" for more information.
- Use available Forms Services utilities and runtime pooling. See Chapter 10.1.3, "Forms Services Utilities" and Chapter 10.2, "Tuning OracleAS Forms Services Applications" for more information.

4.2.1 Using Enterprise Manager Application Server Control to Manage Forms Sessions

By default, Enterprise Manager Application Server Control provides some information about Forms which allows you to centrally modify the configuration files. But you won't experience the full functionality that Enterprise Manager can provide for Forms unless you do the following:

1. In the Forms configuration file (formsweb.cfg) make sure the following variable is set in the default section.

   ```
   em_mode=1
   ```

   This will let Application Server Control show user information for each running Forms application. Only sessions created after setting `em_mode` to 1 will be shown. By default, this value is `0`, which is off.

2. In the Forms configuration file (formsweb.cfg) make sure the following variable is set. You can either set it in the default section or in a specific application section. As with step 1, you can set this variable using Application Server Control:

   ```
   allow_debug=true
   ```

   This will let you turn tracing on and off.

3. **(Windows only)** For the middle tier user that installed Oracle Application Server, you need to give them the "Log on as a batch job" privilege. Logon as either that user, or another user with administrator privileges. Select **Administrative Tools** in the Control Panel. Then select **Local Security Settings** | **Local Policies** | **User Right Assignment**. Add the username of the user who installed Oracle Application Server.

4. **(Windows only)** As the user who installed Oracle Application Server or as a user with administrator privileges, bring up the Windows Services, which can be found

in the Control Panel. Find the Oracle/xxxxxx/ProcessManager service. Right-click it and choose **Properties**. In the Logon tab, make sure **Allow service to interact with desktop** is selected.

5. **(Windows only)** You will need to restart this service. Note that even after it is restarted, it can take up to several minutes for the changes to take effect in Application Server Control.

4.2.2 Configuring Enterprise Manager Grid Control to Manage Forms Services

When you install Forms Services, the Oracle Universal Installer automatically edits Enterprise Manager Grid Control targets.xml file. The targets.xml file contains a list of all the services to be managed by Enterprise Manager.

The first time you use Enterprise Manager to monitor Forms Services, you must perform the following steps for each Forms Services instance to be monitored.

See the Enterprise Manager documentation for information on how to use the Application Server Control Console to access the Enterprise Manager Administration page for a node. (You will need to provide an administrator's username and password.)

To configure Enterprise Manager Grid Control to Manage Forms Services:

1. On the Agent Administration page, all services that are being monitored are listed under the **Agent Monitored Targets** heading.
2. Select the radio button next to the Forms instance to be configured for Enterprise Manager.
3. Click **Edit**.
4. Provide the `ORACLE_HOME` and URL for the Forms instance.
5. Click **OK**.

Note: See the Enterprise Manager help system for more information about other tasks that you can complete on this page.

4.2.3 Accessing Forms Services with Application Server Control Console

To perform most management tasks for a Forms server using Application Server Control Console, you start by navigating to the Forms Home page for the Forms Server in Application Server Control Console.

To navigate to the Forms Home page for a Forms Server in the Application Server Control Console:

1. Using Application Server Control Console, navigate to the home page for the application server that contains Forms server you want to manage.

   For introductory information about using the Enterprise Manager Application Server Control Console, see "Introduction to Administration Tools" in the *Oracle Application Server Administrator's Guide*.

2. In the System Components section on the application server home page, click the link for the Forms server that you want to manage. This displays the Forms home page for the Forms server in the Application Server Control Console.

4.3 Configuring Forms Services

Use the Configuration page in Application Server Control Console to configure Forms Services. This page manages all changes in the formsweb.cfg file for you.

Note: If you manually edit any of the configuration or environment files, you'll need to restart Enterprise Manager as well as restart all Distributed Configuration Management (DCM) processes so that Enterprise Manager can read all changes. If you do not restart Enterprise Manager as well as DCM processes, any changes that you make through Oracle Enterprise Manager 10*g* will overwrite any manual changes you've made to these files. These DCM processes include:

- `emctl stop em`
- `dcmctl stop`
- `opmnctl stopall`
- `opmnctl startall`
- `dcmctl start`
- `emctl start em`

Note: You should backup the formsweb.cfg and default.env files before editing them with Enterprise Manager.

To configure Forms Services:

1. Start the Application Server Control Console.
2. From the Application Server Control Console main page, select the link to the *Oracle Forms Services* instance that you want to configure.
3. From the Forms Services instance, select the **Configuration** tab.
4. Select **Forms Web Configuration** from the **View** pulldown list.
   - To create a new section in the formsweb.cfg file, click **Create New Section** and enter a name for this section on the next page
   - To delete a section in the formsweb.cfg file, click the radio button next to the section to be deleted, then click **Delete** and confirm the deletion on the next page.

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 Application Server Control Console 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 will take longer to save than just the deletion of a single entry.

4.3.1 Configuring Parameters with Application Server Control Console

For a description and the location of the Forms Servlet configuration file (formsweb.cfg), see Chapter 3.2.1.2, "formsweb.cfg".

4.3.1.1 Parameters that Specify Files

The four baseHTML parameters should point to appropriate files. Typically, the following values and their parameters should appear in the default configuration section, as shown in Table 4–1, " Default Configuration Parameters that Specify Files":

Table 4–1 Default Configuration Parameters that Specify Files

Parameter	Value
baseHTML	base.htm
`baseHTMLJinitiator`	basejini.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 `ORACLE_HOME/forms/server`.

4.3.2 Managing Configuration Sections

You create new configuration sections from the **Configuration** tab of Application Server Control Console, which creates the named configurations in the formsweb.cfg file. 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. Start the Enterprise Manager Application Server Control Console.
2. From the Application Server Control Console main page, select the link to the Forms Services instance that you want to configure.
3. From the *Forms Services* instance, select the **Configuration** tab.
4. Click **Create New Section** at the top of the **Configuration** tab.

   The **Forms New Section Name** page appears.
5. Enter a name for your new configuration and click **OK**.
6. If you enter a description of your new section, make sure you save it clicking **Apply** before editing the section and adding parameters.

For example, to create a configuration to run Forms in a separate browser window with a "generic" look and feel, create a new section and add the following parameters from Table 4–2, " Sample Parameters to Add to a New Configuration Section":

Table 4–2 Sample Parameters to Add to a New Configuration Section

Parameter	Value
forms	<module>
separateFrame	True
lookandfeel	Generic

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

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

See Appendix C.1, "Default formsweb.cfg File" for other examples of special configurations.

4.3.2.1 Duplicating a Named Configuration

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

To duplicate a named configuration:

1. Select the radio button next to the section to be duplicated.
2. Click **Duplicate**.
3. On the next page enter a new, unique name for the duplicated section and click **OK**.

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

4.3.2.2 Deleting Named Configurations

When you delete a named configuration, you delete *all* the information within it. If you only want to delete specific parameters, see Section 4.3.3, "Managing Parameters".

To delete a named configuration:

1. Start the Enterprise Manager Application Server Control Console.
2. From the Application Server Control Console main page, select the link to the Forms Services instance that you want to configure.
3. From the Configuration, select the radio button next to the configuration section you want to delete.
4. Click **Delete**.

   The Confirmation page appears.
5. Click **OK**.

   The configuration section is deleted.

   Application Server Control Console returns to the Configuration tab and displays the remaining configurations.

4.3.3 Managing Parameters

Use Application Server Control Console to manage parameters within a named configuration. You can add, edit, or delete parameters from the Edit Section page of Application Server Control Console.

To edit a parameter in a configuration section:

1. From the Configuration tab of Enterprise Manager Application Server Control Console, select the radio button next to the configuration section that contains the parameter that you want to edit.
2. Click **Edit** at the top of this page.

The Edit Section page appears for that selected configuration.

3. Select the radio button next to the parameter you want to edit.
4. Make your changes in the text fields.
5. Click **Apply**.

   Your changes are saved.

To add a parameter to a configuration:

1. From the **Configuration** tab of Application Server Control Console, select the radio button next to the configuration section to which you want to add a parameter.
2. Click **Edit** at the top of this page.

   The Edit Section page appears for that selected configuration.
3. Enter a name and value for the new parameter and click **Add New Parameter**.

   The Edit Section page refreshes and displays the new parameter.
4. Add a description for the new parameter, and click **Apply**.
5. To return to the Forms page, click **Forms** in the breadcrumb trail.

To delete a parameter in a configuration:

1. To edit a configuration section, select the radio button next to it and click **Edit** at the top of this page.

   The Edit Section page appears for the selected configuration.
2. Select the radio button next to the parameter you want to delete.
3. Click **Delete**.
4. Confirm the deletion on the Confirmation page that appears.

   The parameter is deleted from the configuration section.

4.3.4 Default Forms Configuration Parameters

Table 4–3, " System Default Configuration Parameters" describes the default forms configuration parameters in the formsweb.cfg file. For additional information on OracleAS Single Sign-On parameters, see Chapter 6.4, "Enabling OracleAS Single Sign-On for an Application".

These sections include:

- Section 4.3.4.1, "System Default Configuration Parameters"
- Section 4.3.4.2, "Runform parameters (serverArgs parameters)"
- Section 4.3.4.3, "HTML page title, attributes for the BODY tag and HTML to add before and after the form"
- Section 4.3.4.4, "Applet or Object Parameters"
- Section 4.3.4.5, "Parameters for JInitiator"
- Section 4.3.4.6, "Parameters for the Sun Java Plug-in"
- Section 4.3.4.7, "Enterprise Manager Configuration Parameters"
- Section 4.3.4.8, "Oracle Internet Directory Configuration Parameters"

4.3.4.1 System Default Configuration Parameters

These parameters control the behavior of the Forms Servlet. They can only be specified in the servlet configuration file (formsweb.cfg) and cannot be specified as URL query parameters.These parameters are described in Table 4–3, " System Default Configuration Parameters":

Table 4–3 System Default Configuration Parameters

Parameter	Required / Optional	Parameter Value and Description
`baseHTML`	required	The default base HTML file.
`baseHTMLJInitiator`	required	Physical path to HTML file that contains JInitiator tags.
`connectionDisallowedURL`	optional	This is the URL shown in the HTML page that is not allowed to start a new session.
`baseHTMLjpi`	optional	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 Netscape or IE without the IE native settings.
`HTMLdelimiter`	required	Delimiter for variable names. Defaults to %.
`workingDirectory`	required	Defaults to `ORACLE_HOME/forms` if not set.
`envFile`	required	This is set to default.env in the formsweb.cfg file.
`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 (e.g. in the content-type header of a POST). The values of this parameter may be specified either as an IANA character set name (e.g. SHIFT_JIS) or as an Oracle character set name (e.g. 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 is capable of displaying. Also, if the browser allows multibyte characters to be entered directly into a URL, e.g. using the IME, as opposed to URL escape sequences, and if you wish 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 whose name is derived from the value of 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.

Table 4–3 (Cont.) System Default Configuration Parameters

Parameter	Required / Optional	Parameter Value and Description
IE	recommended if there are users with Internet Explorer 5.0 or above browsers	Specifies how to execute the Forms applet under Microsoft Internet Explorer 5.0 or above. If the client is using an Internet Explorer 5.0 or above browser, either the native JVM or JInitiator can be used. A setting of "JInitiator" uses the basejini.htm file and JInitiator. A setting of "Native" uses the browser's native JVM.
log	optional	Supports running and debugging a form from the Builder. Default value is Null.
jvmcontroller	optional	Valid values: See Section 7.5.8.2, "Starting a JVM Controller at the Command Line". In addition, you can specify no JVM. Default value: none 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 (see section 2.3, table 1). If jvmcontroller does not have a value, then the Oracle Forms Runtime Process will start its own in-process JVM, which means that the Java Importer uses pre-10*g* behavior.

4.3.4.2 Runform parameters (serverArgs parameters)

All parameters from here on match variables (%parameterName%) in the baseHTML file. These variables are replaced with the parameter values specified in the URL query string, or failing that, in the formsweb.cfg file. See Chapter 3.3.4, "Specifying Special Characters in Values of Runform Parameters" for information about how runform handles certain special characters that are specified in runform parameter values. These runform parameters are described in Table 4–4, " Runform Parameters (serverArgs Parameters)":

Table 4–4 Runform Parameters (serverArgs Parameters)

Parameter	Required / Optional	Parameter Value and Description
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. For example, a form developed on the Win32 platform may not display properly on the UNIX platform due to varying DPI values. The clientDPI value can be any positive integer. Oracle recommends that you use an integer between 50 and 200, e.g. <param name="clientDPI" value="200">.
escapeparams	optional	Set this parameter to false if you want runform to treat special characters in runform parameters as it did in releases prior to 9.0.4.
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. The default is two minutes. If the heartbeat is less than FORMS_TIMEOUT, the user's session will be kept alive, even if they are not actively using the form.

Table 4–4 (Cont.) Runform Parameters (serverArgs Parameters)

Parameter	Required / Optional	Parameter Value and Description
`form`	required	Specifies the name of the top level Forms module (fmx file) to run.
`userid`	optional	Login string. For example: `scott/tiger@ORADB`.
`otherparams`	optional	This setting specifies command line parameters to pass to the Forms runtime process in addition to `form` and `userid`. Default is: `otherparams=buffer_records=%buffer% debug_ messages=%debug_messages% array=%array% obr=%obr% query_only=%query_only% quiet=%quiet% render=%render% record=%record% tracegroup=%tracegroup% log=%log% term=%term%` 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, in order to provide better control over which runform parameters end users can specify in a URL, use the `restrictedURLparams` parameter.
`debug`	optional	Allows running in debug mode. Default value is `No`.
`buffer`	optional	Supports running and debugging a form from the Builder. Sub argument for `otherparams` Default value is `No`.
`debug_messages`	optional	Supports running and debugging a form from the Builder. Sub argument for `otherparams` Default value is `No`.
`allow_debug`	optional	When set to true, all admin functions from the `forms/frmservlet/admin` screen are activated. `forms/frmservlet/xlate` runs Forms Trace Xlate on a specified trace file. This parameter must be set to `true` before trace logs can be viewed from the User Sessions screen. The default value is `false`; an inline message displays which says that tracing can be viewed only if `allow_debug=true`.
`array`	optional	Supports running and debugging a form from the Builder. Default value is `No`.
`query_only`	optional	Supports running and debugging a form from the Builder. Default value is `No`.
`quiet`	optional	Supports running and debugging a form from the Builder. Default value is `Yes`.
`render`	optional	Supports running and debugging a form from the Builder. Default value is `No`.
`host`	optional	Supports running and debugging a form from the Builder. Default value is `Null`.
`port`	optional	Supports running and debugging a form from the Builder. Default value is `Null`.

Table 4–4 (Cont.) Runform Parameters (serverArgs Parameters)

Parameter	Required / Optional	Parameter Value and Description
record	optional	Supports running and debugging a form from the Builder. Default value is Null.
tracegroup	optional	Supports running and debugging a form from the Builder. Default value is Null.
log	optional	Supports running and debugging a form from the Builder. Default value is Null.
term	optional	Supports running and debugging a form from the Builder. Default value is Null.
em_trace	For internal use only.	

4.3.4.3 HTML page title, attributes for the BODY tag and HTML to add before and after the form

For security reasons these parameters may not be set using URL query parameters, as described in Table 4–5, " HTML Page Parameters":

Table 4–5 HTML Page Parameters

Parameter	Required / Optional	Parameter Value and Description
pageTitle	optional	HTML page title, attributes for the BODY tag, and HTML to add before and after the form.
HTMLbodyAttrs	optional	Attributes for the <BODY> tag of the HTML page.
HTMLbeforeForm	optional	HTML content to add to the page above the area where the Forms application will be displayed.
HTMLafterForm	optional	HTML content to add to the page below the area where the Forms application will be displayed.

4.3.4.4 Applet or Object Parameters

The following parameters in Table 4–6, " Applet or Object Parameters" are specified in the baseHTML file as values for object or applet parameters. For example: `<PARAM NAME="serverURL" VALUE="%serverURL%">`

Table 4–6 Applet or Object Parameters

Parameter	Required / Optional	Parameter Value and Description
serverURL	required	/forms/lservlet (see Chapter 1.5, "Forms Listener Servlet".)
codebase	required	Virtual directory you defined to point to the physical directory ORACLE_HOME/forms/java, where, by default, the applet JAR files are downloaded from. The default value is /forms/java.
imageBase	optional	Indicates where icon files are stored. Choose between: ■ 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 default. In deployments that make use of the Forms Server CGI, you must specify the icon path in a custom application file.
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
restrictedURLparams	optional	Specified by an administrator to restrict a user from using certain parameters in the URL. If the number of parameters is more than one, then they should be separated by a comma. The restrictedURLparams itself cannot be the value of this parameter i.e., restrictedURLparams. Default value is HTMLbodyAttrs,HTMLbeforeForm, pageTitle,HTMLafterForm,log,allow_ debug,allowNewConnections
formsMessageListener	optional	Forms applet parameter.
recordFileName	optional	Forms applet parameter.
width	required	Specifies the width of the form applet, in pixels. Default is 650.
height	required	Specifies the height of the form applet, in pixels.Default is 500.
separateFrame	optional	Determines whether the applet appears within a separate window. Legal values: True or False.
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).
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.
lookAndFeel	optional	Determines the applications look-and-feel. Legal values: Oracle or Generic (Windows look-and-feel).
colorScheme	optional	Determines the application's color scheme. Legal values: Teal, Titanium, Red, Khaki, Blue, Olive, or Purple. **Note**: colorScheme is ignored if lookAndFeel is set to Generic.

Table 4–6 (Cont.) Applet or Object Parameters

Parameter	Required / Optional	Parameter Value and Description
serverApp	optional	Replace default with the name of your application file (if any). Use application classes for creating application-specific font mapping and icon path settings. To set the parameter include the file name if file is in ORACLE_HOME/forms/java/oracle/forms/registry or include the virtual path and file name.
archive	optional	Comma-separated list of archive files that are used when the browser detected is neither Internet Explorer using native JVM nor JInitiator. (The default is frmall.jar.) To set the parameter include the file name if the file is in the codebase directory or include the virtual path and file name.
archive_jinit	optional	Comma-separated list of JAR file(s) that is used when the browser detected is JInitiator. (The default is frmall_jinit.jar.) To set the parameter include the file name if the file is in the codebase directory or include the virtual path and file name.
archive_ie	optional	Comma-separated list of CAB file(s) that is used when the browser detected is Internet Explorer using native JVM. (The default is frmall.cab.)
networkRetries	optional	In situations of high load or network failures, you can specify the number of times (up to 10) the client will attempt to send a request to the intended servlet engine. The default setting is 0, in which case the Forms session will terminate after one try.
mapFonts	optional	<PARAM NAME = "mapFonts" VALUE = "yes" > to trigger font mapping. As a result of some font rendering code changes in JDK 1.3, the font heights set in JDK 1.1 increased in JDK 1.3. As this may cause display issues, you can map the JDK 1.3 fonts so that the font sizes are the same as they were in JDK 1.1.

4.3.4.5 Parameters for JInitiator

The following parameters are specific to JInitiator, as described in Table 4–7, " Parameters for JInitiator":

Table 4–7 Parameters for JInitiator

Parameter	Required / Optional	Parameter Value and Description
jinit_download_page	required (Netscape only)	If you create your own version of the Jinitiator download page, set this parameter to point to it. Default is /forms/jinitiator/us/JInitiator/jinit.download.htm.
jinit_classid	required (IE only)	Default is clsid:CAFECAFE-0013-0001-0017-ABCDEFABCDEF
jinit_exename	required	Default is jinit.exe#Version=1.3.1.17
jinit_mimetype	required (Netscape only)	Default is application/x-jinit-applet;version=1.3.1.17
baseHTMLJInitiator	required	Physical path to HTML file that contains JInitiator tags.

4.3.4.6 Parameters for the Sun Java Plug-in

The following parameters are for use with the Sun Java Plug-in, as described in Table 4–8, " Parameters for Sun's Java Plug-in":

Table 4–8 Parameters for Sun's Java Plug-in

Parameter	Required / Optional	Parameter Value and Description
`jpi_codebase`	required	Sun's Java Plug-in codebase setting
`jpi_classid`	required	Sun's Java Plug-in class id
`jpi_download_page`	required	Sun's Java Plug-in download page

4.3.4.7 Enterprise Manager Configuration Parameters

The following parameters are for configuring Enterprise Manager, as described in Table 4–9, " Enterprise Manager Configuration Parameters":

Table 4–9 Enterprise Manager Configuration Parameters

Parameter	Required / Optional	Parameter Value and Description
`em_mode`	required	1 is to enable. 0 is to disable. 1 indicates that all Enterprise Manager information is available, including metrics and servlet status. 0 indicates that only configuration information is available.
EndUserMonitoringEnabled	Optional	Indicates whether End User Monitoring integration is enabled.
EndUserMonitoringURL	Optional	indicates where to record End User Monitoring data.

4.3.4.8 Oracle Internet Directory Configuration Parameters

The following parameters are for configuring Oracle Internet Directory, as described in Table 4–10, " Oracle Internet Directory Configuration Parameters". You can only configure these parameters if you are using an OracleAS Infrastructure instance.

Table 4–10 Oracle Internet Directory Configuration Parameters

Parameter	Required / Optional	Parameter Value and Description
`oid_formsid`	required	Configured during the OracleAS installation, so you do not need to change this.
`ORACLE_HOME`	required	Configured during the OracleAS installation, so you do not need to change this.

4.4 Configuring Environment Variables with Enterprise Manager

Use the **Environment** tab of the Enterprise Manager Application Server Control Console page to manage Environment Variables. From this page, you can add, edit, or delete environment variables as necessary.

The environment variables such as `PATH`, `ORACLE_HOME`, and `FORMS_PATH` for the Forms runtime executable (`frmweb.exe` on Windows and `frmweb` on Solaris) are defined in the **Environment** tab. The Listener Servlet calls the executable and initializes it with the variable values provided in the environment file, which is `ORACLE_HOME/forms/server/default.env` by default.

Any environment variable that is not defined in that page is inherited from the servlet engine (OC4J). The environment file must be named in the `envFile` parameter in the Default section of the Forms 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 will need administrator privileges to alter registry values.
- You do not need to restart the server for configuration changes to take effect.
- Environment variables not set in the environment file or Windows registry are inherited from the environment of the parent process, which is the servlet engine (OC4J).

Note: You cannot create or delete environment files through Enterprise Manager Application Server Control Console. Environment files must be created manually in `ORACLE_HOME/forms/server` with a `.env` extension.

Likewise, environment files cannot be deleted from Application Server Control. For a new environment file to be picked up by Application Server Control Console and for a deleted one to disappear you will need to restart the Enterprise Manager processes:

- `emctl stop em`
- `emctl start em`

Table 4–11, " Default Environment Variables" describes important environment variables that are specified in default.env:

Table 4–11 Default Environment Variables

Environment Variable	Valid Values	Purpose
`ORACLE_HOME`	`ORACLE_HOME` (default)	Points to the base installation directory of any Oracle product.
`PATH`	`ORACLE_HOME\bin` (default)	Contains the executables of Oracle products.
`FORM_PATH`	`ORACLE_HOME\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 **Solaris**, separate paths with a *colon* (:).

Table 4–11 (Cont.) Default Environment Variables

Environment Variable	Valid Values	Purpose
FORMS_TIMEOUT	Default: 15 Valid Values: 3 – 1440 (1 day) Example: FORMS_TIMEOUT=1440	This parameter specifies the amount of time in elapsed minutes before the Form Services process is terminated when there is no client communication with the Form Services. Client communication can come from the user doing some work, or from the Forms Client heartbeat if the user is not actively using the form.
TNS_ADMIN	ORACLE_HOME/network/admin	Specifies the path name to the TNS files such as TNSNAMES.ORA, SQLNET.ORA etc.
CLASSPATH	ORACLE_HOME/jdk/bin/java	Specifies the Java class path, which is required for the Forms debugger.
REPORTS_CLASSPATH	ORACLE_ HOME/jlib/zrclient.jar:ORACLE_ HOME/reports/jlib/rwrun.jar	This setting is only needed if Reports applications are called from Forms applications
REPORTS_SERVERMAP	cluster:repserver Where: cluster is the Reports Server cluster name; repserver is the Reports Server name.	This setting is needed if Forms applications are calling Reports applications from a Reports cluster instead of a Reports server. This setting is also needed when a Forms application calls a Reportss application using web.show_document. See *Oracle Application Server Reports Services Publishing Reports to the Web* for additional Reports configuration information.
ORACLE_ GRAPHICS6I_HOM	<GRAPHICS6I_HOME>	These settings are only needed if Graphics applications are called from Forms applications Use Enterprise Manager to set the ORACLE_HOME value to use Graphics applications.

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.

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

4.5 Managing User Sessions

Oracle Application Server Forms Services contains features to help administrators manage user sessions, including:

- Section 4.5.1, "Allowing New Users Sessions"
- Section 4.5.2, "Disabling New User Sessions"
- Section 4.5.3, "Terminating a User Session on a Forms Services Instance"

4.5.1 Allowing New Users Sessions

By default, users can create a new Forms session, which is indicated by the green traffic light. You can also enable users to create Forms sessions after you've disabled them.

To allow new Forms User sessions:

- From the Enterprise Manager Oracle Application Server Forms Services Overview page, click **Enable** (default).

  The traffic light changes to green

4.5.2 Disabling New User Sessions

To disable new user Forms user sessions:

- From the Enterprise Manager Oracle Application Server Forms Services Overview page, click **Disable**.

  The traffic light changes to yellow.

When you press **Disable**, a new parameter is added to the default section of the formsweb.cfg file. The parameter is called `allowNewConnections` and pressing **Disable** sets it to `false`. When new user sessions are disabled, attempted connections will be directed to a URL identified by the formsweb.cfg parameter `connectionDisallowedURL` (default section) e.g:

```
connectionDisallowedURL=www.oracle.com
connectionDisallowedURL=http://www.oracle.com
```

If no `connectionDisallowedURL` is specified then the following message will be displayed in the browser:

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

However, when you disable new user sessions, existing forms sessions are unaffected and the OC4J instance remains up.

4.5.3 Terminating a User Session on a Forms Services Instance

1. Start the Oracle Enterprise Manager 10*g* Application Server Control Console.
2. Select the link to the Forms Services instance that has the user session to be terminated.
3. From the **Overview** page for the Forms Services instance, select the **Session Details** link.
4. Click the radio button next to the user session to be deleted.
5. Click **Stop**.

6. The Confirmation page appears.
7. Click **Yes**.

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

4.6 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, i.e. formsweb.cfg. The parameter that is set in the formsweb.cfg can be overridden by the parameter set in the URL.

Note: You manage the `restrictedURLparams` parameter through the Configuration page of Enterprise Manager Application Server Control Console.

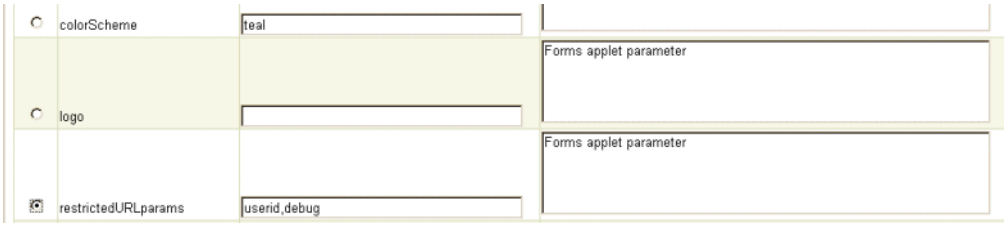
A Forms administrator can override this default behavior, and give the Forms administrator full control over what parameter can be used in the URL.

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 usages 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 would like to disable 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.

Figure 4–1, "Defining the restricedURLparams Parameter" is an example of how the `restrictedURLparams` parameter is defined in the [myApp] section to override the one set in the [default] configuration section:

Figure 4–1 Defining the restricedURLparams Parameter



By default, this user, `scott`, is not allowed to debug this Forms application, use Forms Trace, or edit records in it. In the myApp section, user scott is only forced to log in when accessing the application, and not allowed to debug it. He can now, though, work with Forms Trace and edit records through a URL for this application.

An administrator can use the `restrictedURLparams` parameter to redirect a user to an error page that lists the parameters the user is restricted from using (or allowed to use) for this application.

4.6.1 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 will locate `[myApp]` in the formsweb.cfg file and launch 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 allows you to test your Oracle Forms Services installation and configuration. Thus if you don't specify an application, Forms will launch the test.fmx file. You could change this to:

```
form=
```

and the form will 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. You'll need to change some parameters in the formsweb.cfg file.

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 JInitiator client
baseHTMLjinitiator=basejini.htm
# System parameter: base HTML file for use with Sun's Java Plug-In
baseHTMLjpi=basejpi.htm
# System parameter: base HTML file for use with Microsoft Internet Explorer
# (when using the native JVM)
baseHTMLie=baseie.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_HOME/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 redirecting of client information and presenting a message page instead is *not* the same Web 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 JInitiator client
#baseHTMLjinitiator=basejini.htm
baseHTMLjinitiator=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)
#baseHTMLie=baseie.htm
baseHTMLie=forbidden.html
```

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 as long as 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
baseHTMLjinitiator=basejini.htm
baseHTMLjpi=basejpi.htm
baseHTMLie=baseie.htm
```

If you don't specify these base HTML values, and when a user runs an application, they will see the forbidden.html page because the application-specific configuration section hasn't overridden the default values.

4.7 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, simply 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.

4.8 Including Graphics in Your Oracle Forms Application

In order to integrate graphics applications with your Oracle Forms applications, you must set the path definition in the Forms Servlet environment to include graphics as follows:

```
PATH=ORACLE_HOME/bin;<GRAPHICS6I_HOME>/bin
```

The path definition of the Forms Servlet environment, is taken from the path definition of the servlet container. The file or location where the path will be defined is different for different servlet containers.

For more information about graphics, see *Oracle Forms Developer and Oracle Application Server Forms Services: Migrating Forms Applications from Forms6i* and *Deploying Graphics in Oracle9iAS Forms Services*, available at Oracle Technology Network (OTN), `http://www.oracle.com/technology/products/forms/`.

4.8.1 Oracle Graphics 6*i* and Oracle Database 9.0.1.4.0 (64bit)

Due to a limitation in the 8.0.6 RSF, Oracle Graphics 6*i* on Windows cannot connect to a 64-bit database. Thus, if you are using Oracle Forms 10*g* (9.0.4) or later to connect to a 64-bit database, and want to integrate with Oracle Graphics, you will need to upgrade your Oracle 6*i* Home (where Graphics is installed) to include an RSF version that contains a fix to bug 3088708. Please contact Oracle Support regarding availability of such an RSF.

4.8.2 Configuring Graphics 6*i* for use by Reports Server

Perform the following to correctly setup Reports/Graphics for Forms/Reports/Graphics integration:

1. In the `graphicsrun.sh` script enter the following:

   ```
   ORACLE_GRAPHICS6I_HOME=<location forms6i>
   export ORACLE_GRAPHICS6I_HOME
   TK_PRINTER=<real printer>
   ```

2. In the reports.sh script enter the following:

   ```
   ORACLE_GRAPHICS6I_HOME=<location forms6i>; export ORACLE_GRAPHICS6I_HOME
   REPORTS_DEFAULT_DESPLAY=NO; export REPORTS_DEFAULT_DESPLAY
   DISPLAY=<computer name>:0.0; export DISPLAY
   ```

4.9 Deploying Icons and Images Used by Forms Services

This section explains how to specify the default location and search paths for icons and images in Registry.dat.

4.9.1 Managing Registry.dat with Application Server Control

Use Application Server Control to change, add, or delete parameters from Registry.dat.

To change a Registry.dat parameter value:

1. Select the Configuration page of Enterprise Manager.
2. From the View dropdown list, select **Forms Font and Icon Mapping (Registry.dat)**.
3. Select a radio button next to a parameter and change the value(s) for it in the **Value** text field.
4. Click **Apply**.

   Your changes are saved.

To add a Registry.dat parameter and its value:

1. Select the Configuration page of Enterprise Manager.
2. From the View dropdown list, select **Forms Font and Icon Mapping (Registry.dat)**.
3. At the bottom of the Registry.dat page, enter a name for the parameter in the **Name** text field.
4. Enter a value for this new parameter in the **Value** text field.
5. Click **Add New Parameter**.

   Your changes are saved.

To delete a Registry.dat parameter and its value:

1. Select the Configuration page of Enterprise Manager.
2. From the **View** dropdown list, select **Forms Font and Icon Mapping (Registry.dat)**.
3. Select a radio button next to a parameter and click **Delete**.
4. The Confirmation page appears, click **Yes**.
5. The parameter is deleted and the Configuration page reappears.

4.9.2 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 Forms Web Configuration page of Enterprise Manager Application Server Control Console. Alternatively, you can change the `default.icons.iconpath` value of the Registry.dat file in the `forms/java/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 if you want to store images in a central location independent of any application and independent of the Oracle Forms installation.

4.9.2.1 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 via 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 needs to 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 (for example, `archive_jini=frmall_jinit.jar, myico.jar`). Now, when the initial application starts, the icon files are downloaded to and permanently stored on the client until the archive file is changed.

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

4.9.2.2 Adding Icon Changes to Registry.dat

If you want to add icon changes to the Registry.dat file used by your application, it is recommended that you make a copy of the existing Registry.dat file and edit the copied file.

To create a copy of the Registry.dat file:

1. Copy the Registry.dat text file found in the `ORACLE_HOME/forms/java/oracle/forms/registry` directory to another directory. This directory must be mapped to a virtual directory for your Web server (for example, `/appfile`).
2. Rename this new file (for example, myapp.dat).
3. Modify the `iconpath` parameter specifying your icon location:

   ```
   default.icons.iconpath=/mydir or http://myhost.com/mydir
   ```
   (for an absolute path)

   or

   ```
   default.icons.iconpath=mydir
   ```

   (for a relative path, starting from the `DocumentBase` Directory)
4. Modify the `iconextension` parameter:

   ```
   default.icons.iconextension=gif
   ```

   or

   ```
   default.icons.iconextension=jpg
   ```

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 4–12, " Icon Location Guide" describes the correct locations where to place your application icons:

Table 4–12 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.
Registry.dat	Applications with custom icons that are stored in a different location as the Oracle Forms install (can be another server). Useful if you need to make other changes to the Registry.dat file like font mapping.	Copy Registry.dat and change ServerApp parameter in formsweb.cfg.

4.9.3 SplashScreen 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 Forms 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.

4.9.4 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.

4.9.4.1 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`.

4.9.4.2 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.

If you want 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. It is the default behavior.
- **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
```

4.9.5 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 Web configuration or HTML file (for the images).
- What you specify in the `imageBase` parameter in the Forms Web Configuration page of Application Server 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.

4.9.5.1 DocumentBase

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

Table 4–13 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)
`iconpath=/mydir` (specified in your application file)	`http://host/mydir` (absolute path)

Table 4–14 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)
file.gif (specified, for example, in formsweb.cfg as `splashscreen=file.gif`)	`http://host/documentbase/file.gif`

4.9.5.2 CodeBase

Use the `imageBase=CodeBase` parameter to enable the search of the icons and images in a Jar file:

Table 4–15 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 4–16 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.

4.10 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.

4.10.1 Specifying Language Detection

For each configuration section in the Forms 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
```

4.10.2 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 (e.g. 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: basejini.htm (Oracle Forms) -->
 <BODY>
 ...
 <OBJECT classid=...
>
<PARAM NAME="inlineIME" VALUE="false">
<EMBED SRC="" ...
inlineIME="false"
>
...
.
</BODY>
</HTML>
.
To have inline IME support, forms client needs to be Jinitator 1.3.1 or Plug-in
1.4.1+.
```

For more information about using baseHTML, see Appendix C.3, "base.htm, basejini.htm, and basejpi.htm Files".

4.10.3 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 will look for a language-specific configuration section matching the first language. If one is not found, it will look for the next and so on. If no language-specific configuration is found, it will use 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 will look for a language-specific section in the default section matching the first language (for example, [.fr]).

4.10.3.1 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.

4.11 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_HOME/admin/resource/<language directory>` directory in UNIX, for example `ORACLE_HOME/forms/admin/resource/US`. For Windows, the location is `ORACLE_HOME\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 a different resource file will be used. e.g. `NLS_LANG=GERMAN_GERMANY=WE8ISO8859P1 fmrwebd.res` file will be 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/forms/f90servlet?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
```

4.11.1 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 6*i* 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 Application Server Control.

4.11.1.1 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` e.g to `fmrweb_orig.res`, and copy `fmrpcweb.res` to `fmrweb.res`. Alternatively use the term parameter as described above.

4.11.1.2 Exceptions/ Special Key Mappings

The following examples show special key mappings:

- Section 4.11.1.2.1, "Mapping F2"
- Section 4.11.1.2.2, "Mapping for ENTER to Fire KEY-ENTER-TRIGGER"
- Section 4.11.1.2.3, "Mapping Number Keys"
- Section 4.11.1.2.4, "Mapping for ESC Key to exit out of a Web Form"

4.11.1.2.1 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 : 3" 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 in order 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.

4.11.1.2.2 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 will fire the Key-Enter trigger.

4.11.1.2.3 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'

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) e.g:

   ```
   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"
   ```

4.11.1.2.4 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 will exit.

5

Using OracleAS Forms Services with the HTTP Listener and OC4J

Oracle Application Server Containers for J2EE (OC4J) is a complete J2EE (Java 2 Platform Enterprise Edition) server written entirely in Java that executes in a standard Java Runtime Environment (JRE). It provides a complete J2EE environment that contains, among other things, an OC4J Web container.

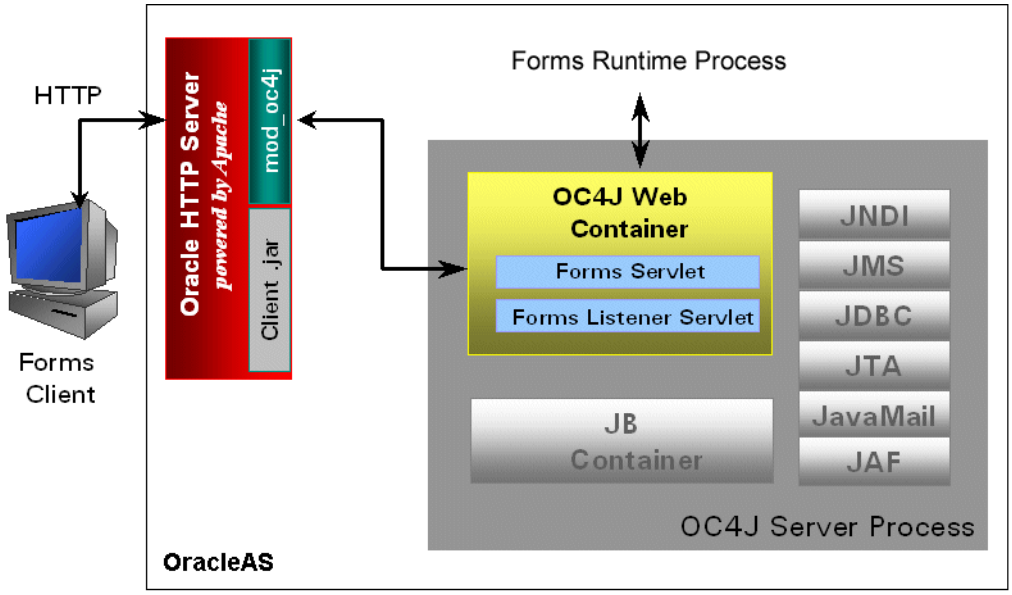
This chapter contains the following sections:

5.1 OC4J Server Process

In a simple scenario, 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 OC4J. The Forms Listener Servlet establishes a Forms Server 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 will establish 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 Section 5.2, "Performance/Scalability Tuning").

Figure 5–1, "OC4J Architecture and Forms Services" shows how Forms Services uses the OC4J architecture:

Figure 5–1 OC4J Architecture and Forms Services



5.2 Performance/Scalability Tuning

The steps for tuning the Forms Listener Servlet are similar to steps for tuning any high throughput servlet application. You will have to take into account resource management and user needs for optimal tuning of your particular Forms Services configuration. For more information, see *Oracle Application Server Performance Guide*, available on Oracle Application Server Disk 1 CD or OTN at `http://www.oracle.com/technology/documentation/`.

5.3 Limit the number of HTTPD processes

To avoid spawning too many HTTPD processes (which is memory consuming) set the following directive in the Oracle HTTP Listener configuration file (httpd.conf):

```
KeepAlive Off
```

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.

5.4 Set the MaxClients Directive to a High value

You can let the HTTP Listener determine when to create more HTTPD daemons. 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` means 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.

5.5 Load Balancing OC4J

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 OC4J 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 the OC4J 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 OC4J chapter in the *Oracle Application Server Performance Guide* (available on Oracle Application Server Disk 1 CD or OTN at `http://www.oracle.com/technology/products/ias/`.

For more Forms-specific information, see the *Oracle Developer Suite and Oracle Application Server Release Notes.*

The following images illustrate four possible deployment scenarios:

- Figure 5–2: Balancing incoming requests between multiple OC4J engines on the same host as the Oracle HTTP Listener.
- Figure 5–3: Balancing incoming requests between multiple OC4J engines on a different host to the Oracle HTTP Listener.
- Figure 5–4: Balancing incoming requests between multiple OC4J engines on multiple different hosts and multiple different hosts each running an Oracle HTTP Listener.
- Figure 5–5: Balancing incoming requests between multiple OC4J engines on a single host but with multiple different hosts each running an Oracle HTTP Listener.

Figure 5–2 Multiple OC4J engines on the same host as the Oracle HTTP Listener

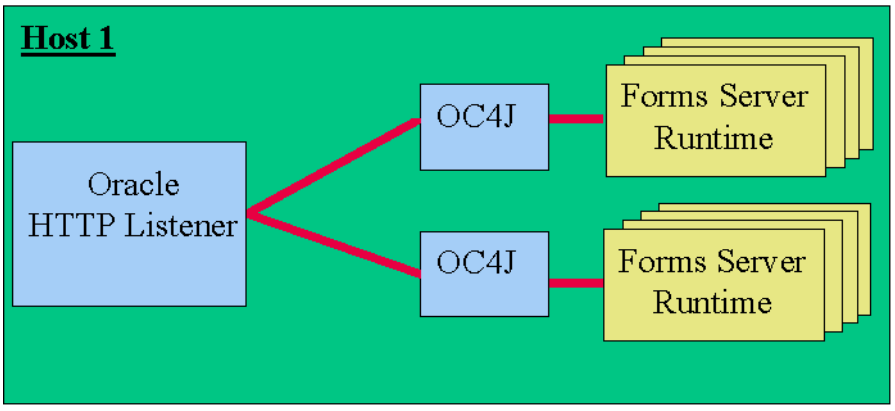


Figure 5–3 Multiple OC4J engines on a different host to the Oracle HTTP Listener

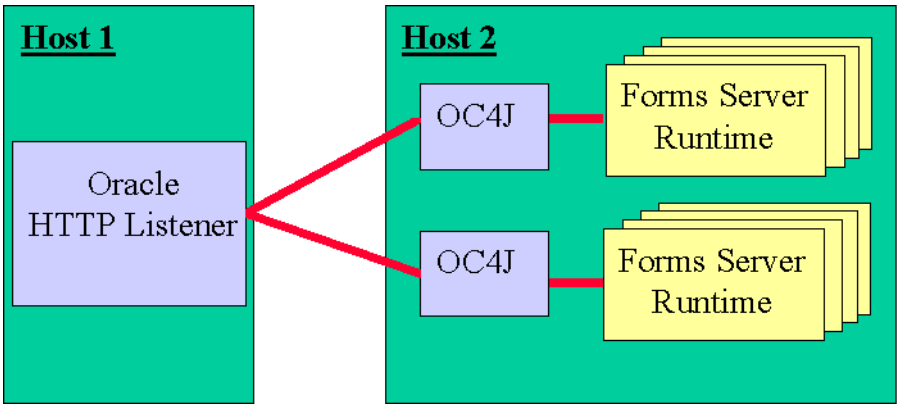


Figure 5–4 Multiple OC4J engines and multiple Oracle HTTP Listeners on different hosts

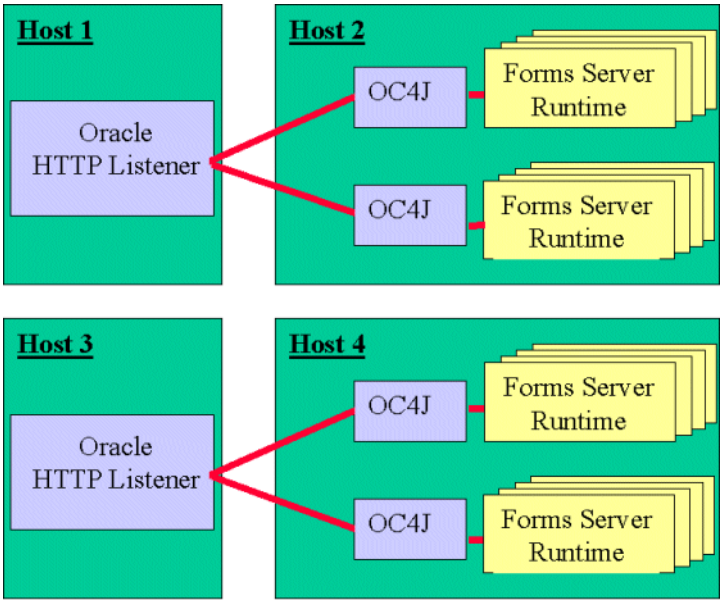
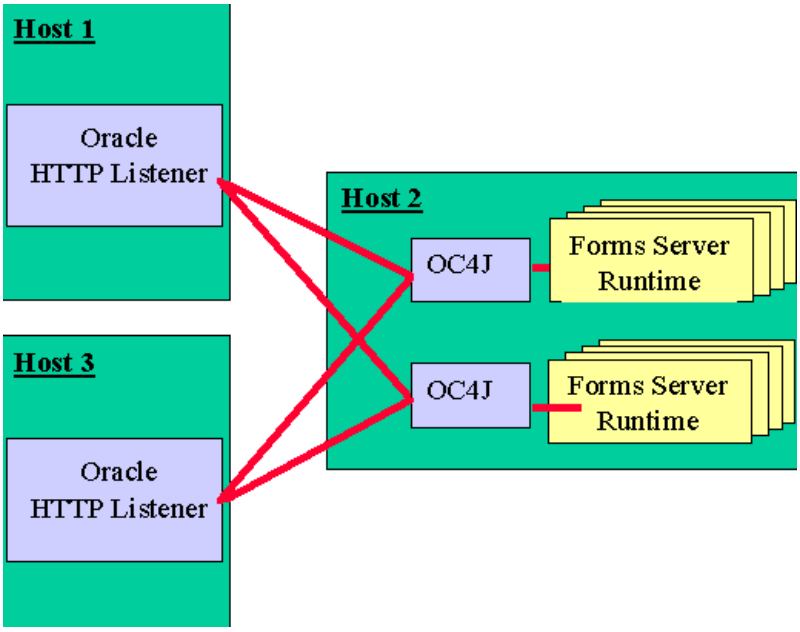


Figure 5–5 Multiple Oracle HTTP Listeners on different hosts with multiple OC4J engines on one host



For more information about tuning and optimizing Forms Services with the HTTP Listener and OC4J, see *Oracle Application Server Performance Guide*, available on Oracle Application Server Disk 1 CD or Oracle Technology Network (OTN) at `http://www.oracle.com/technology/products/ias/`.

5.6 Using HTTPS with the Forms Listener Servlet

Using HTTPS with Oracle Forms is no different than using HTTPS with any other Web-based application.

5.7 Server Requirements

HTTPS requires the use of digital certificates. Because Oracle Application Server 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.

5.8 Client Requirements: Using HTTPS with Oracle JInitiator

If your end users are running Oracle JInitiator as the Web browser JVM, then you need to check that the Root Certificate Authority of your Web site's SSL certificate is one of those defined in the JInitiator **certdb.txt** file.

The certdb.txt file is usually found under `c:\program files\oracle\jinitiator <version>\lib\security` on the computer where JInitiator was installed.

Note: If you are running with Oracle Application Server Web Cache enabled (which is usually the case), you should use the file `OracleAS_HOME/webcache/wallets/default/b64certificate.txt`. If you are not running with Web Cache (that is, you are accessing the Oracle HTTP Server directly) you will need to create the demo root certificate file as follows:

1. Start Oracle Wallet Manager
2. Open `ORACLE_HOME/Apache/Apache/conf/ssl.wlt/default/ewallet.p12`
3. Select menu option **Export Wallet** under the **Operations** menu
4. Save as text file "demoCertCA.txt"

Once you have the required certificate file, you should follow the instructions to configure JInitiator to use the certificate (appending it to JInitiator's certdb.txt file).

For more information about Oracle JInitiator, see Appendix B, "JInitiator".

5.9 Using the Hide User ID/Password Feature

With Oracle Application Server Forms Services, the `userid` parameter value is not included in the HTML generated by the Forms Servlet.

By default, this feature enables Forms Services to:

- Specify the user/password@database using a parameter called “userid” (not case-sensitive). This is already done if you are using the default baseHTML files, which are provided when Oracle Forms is installed. They contain syntax like "userid=%userid%".
- Use the Forms Servlet rather than static HTML files.

5.10 Using an Authenticating Proxy to Run Oracle Forms Applications

The default configuration as set up by the Oracle Application Server 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.

If users are running Netscape with JInitiator, there are certain configuration requirements necessary to ensure that the proxy’s authentication cookie gets sent with all requests to the server. The basic requirement is that every URL that JInitiator has to access (for the Jar files *and* for the Forms Listener Servlet) MUST be under the document base of the HTML page. This is achieved by using the Forms Servlet to generate the page, invoking it using a URL under `/forms`, such as `https://myserver.com/forms/frmservlet?config=myApp`.

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

5.11 Oracle Forms Services and SSL

By default, the HTTPS port is not enabled when installing Oracle Application Server 10*g*. There are two steps to enable SSL: enable the HTTPS port in Oracle HTTP Server, then enable Web Cache to accept HTTPS connections from Oracle HTTP Server.

> **Note:** If you've coded your Forms application such that the logon dialog appears (because you haven't specified the user/password as part of the configuration for the application you're running) and you're not running your application with SSL/HTTPS, you should be aware that there is a potential security issue. The password that is entered in the logon dialog will be sent across the network.

5.11.1 Configuring Oracle HTTP Server to use SSL

When you enable Oracle HTTP Server to use SSL, you modify a portion of the Oracle Process Manager and Notification Server (OPMN) configuration file. After you've modified and saved this file, you will need to restart OPMN processes.

To configure Oracle HTTP Server to use SSL:

1. Open `ORACLE_HOME/opmn/conf/opmn.xml` in a text editor and find this block of code:

```
<ias-component id="HTTP_Server">
   <process-type id="HTTP_Server" module-id="OHS">
          <module-data>
                 <category id="start-parameters">
                        <data id="start-mode" value="ssl-disabled"/>
                 </category>
          </module-data>
   <process-set id="HTTP_Server" numprocs="1"/>
</process-type>
</ias-component>
```

2. Change the `start-mode` parameter value to `ssl-enabled`:

```
<ias-component id="HTTP_Server">
   <process-type id="HTTP_Server" module-id="OHS">
          <module-data>
                 <category id="start-parameters">
                        <data id="start-mode" value="ssl-enabled"/>
                 </category>
          </module-data>
   <process-set id="HTTP_Server" numprocs="1"/>
</process-type>
</ias-component>
```

3. Force OPMN to reload the modified opmn.xml configuration file:

```
opmnctl reload
```

5.11.2 Configuring Oracle Web Cache to use SSL

Use the Web Cache Admin page to enable HTTPS connections from Oracle HTTP Server.

To configure Web Cache to use SSL:

1. Open the Web Cache Manager page. If you configured OracleAS Web Cache during installation, you can access it as `http://hostname.domain:port`. Its default port is 4000, or as the Web Cache HTTP Listen port number as listed in:
   - Solaris: `ORACLE_HOME/install/portlist.ini`
   - Windows: `ORACLE_HOME\install\portlist.ini`
2. Login as the application server administrator.
3. Locate the **Port** section in the navigator frame and click **Listen Ports**.
4. Click **Add**.
5. From the For Cache dropdown list, select the target Web Cache.
6. Enter the following information, as shown in Table 5–1:

Table 5–1 HTTPS Port Configuration Information

Setting	Description
IP Address	Any valid IP address
Port Number	443
Protocol	HTTPS
Require Client-Side Certification	Enable or disable client-side certificates. Select Require Client-Side Certificate to enable OracleAS Web Cache to require browsers to provide SSL certificates. You'll need to import ewallet.p12 file from `webcache/wallet/default`. A client-side certificate is a method for verifying the identity of the client. It binds information about the client user to the user's public key and must be digitally signed by a trusted certificate authority.
Wallet	Enter the directory location of the wallet. This directory must contain an existing wallet. This wallet is used for administration, invalidation, and statistics monitoring of HTTPS requests for sites hosted by OracleAS Web Cache. Oracle recommends entering the location, even if the default is being used. The default location is `ORACLE_HOME/webcache/wallets/default`.

> **Note:** When selecting and using client-side certification, you must use Sun Java Plug-in 1.4.2 or later. Visit `http://java.sun.com/` for more information.

7. Click **Apply Changes**.
8. Restart Web Cache.

5.11.3 Running a Form with SSL

Running a Forms application that uses an HTTPS port requires a certificate to be imported. If you access Web Cache through port 4443, you need to import the Web Cache certificate. If you access Oracle HTTP Server through port 4444, you need to import the Oracle HTTP Server certificate.

To import the Web Cache certificate:

1. Open ORACLE_HOME\webcache\wallets\default
   - Windows: Invoke Wallet manager launch.exe
   - Solaris: owm
2. Open `ORACLE_HOME\WebCache\wallets\default`.
3. Enter `welcome` as the password.
4. Select **Auto Login**.
5. Select **FOR TEST PURPOSES ONLY**.
6. Choose **[Operations]-[Export Trusted Certificate]** and provide a name.
7. Open this file in a text editor and copy all of its contents and append it to `C:\Program Files\Oracle\JInitiator 1.3.1.21\lib\security\certdb.txt`.
8. Run the Form Servlet as `https://computer.mycompany.com:4443/forms/frmservlet`.
9. Verify that the JInitiator log window shows the HTTPS protocol.

To import the Oracle HTTP Server Certificate:

1. Invoke Wallet manager:
   - Windows: Invoke Wallet manager launch.exe
   - Solaris: owm
2. Open `ORACLE_HOME\Apache\Apache\conf\ssl.wlt\default`.
3. Enter `welcome` as the password.
4. Select **Auto Login**.
5. Select **FOR TEST PURPOSES ONLY**.
6. Choose **[Operations]-[Export Trusted Certificate]** and provide a name.
7. Open this file in a text editor and copy all of its contents and append it to `C:\Program Files\Oracle\JInitiator 1.3.1.21\lib\security\certdb.txt`.

8. Run the Form Servlet as `https://computer.mycompany.com:4444/forms/frmservlet`.
9. Verify that the JInitiator log window displays the HTTPS protocol.

5.11.4 Configuring SSL with a Load Balancing Router

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

Follow these steps to enable SSL with your Forms applications over a load balancing router:

1. Open the **Security Alert** dialog by opening `https://mycomputer.us.oracle.com:443/forms/frmservlet` in a Web browser.
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. Open `c:\temp\forms.cer` in a text editor.
12. Copy the contents of the file into JInitiator's certdb.txt file in `lib\security`.
13. Save `certdb.txt`.
14. Reopen the Security Alert dialog and click **Yes**.

6

Using Forms Services with Oracle Application Server Single Sign-On

This chapter contains the following sections:

6.1 Overview

Oracle Application Server Single Sign-On enables an application to authenticate users by means of a shared authentication token or authentication authority. For example, a user authenticated for one application is automatically authenticated for all other applications within the same authentication domain.

Oracle Application Server Forms Services applications can run in a Single Sign-on environment using Oracle Single Sign-On Server 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.

> **Note:** Single Sign-on is not available with the Oracle Application Server Forms and Reports Services installation type. See the *Oracle Application Server Forms and Reports Installation Guide* for information on how to use an OracleAS Infrastructure.

The OracleAS Single Sign-On Server can be used to enable OracleAS Single Sign-On for other applications that are not Oracle products, like, for example, custom built J2EE applications.

Oracle Forms applications seamlessly integrate into a company's OracleAS Single Sign-On architecture based on Oracle Single Sign-On Server and the Oracle Internet Directory. Oracle Application Server 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.

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

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

- Section 6.2.1, "Dynamic Resource Creation When A Resource Is Not Found In Oracle Internet Directory"
- Section 6.2.2, "Support for Default Preferences in Oracle Internet Directory to Define Forms Resources"
- Section 6.2.3, "Support for Dynamic Directives With Forms and OracleAS Single Sign-On"
- Section 6.2.4, "Support for Database Password Expiration for Forms Running with OracleAS Single Sign-On"

6.2.1 Dynamic Resource Creation When A Resource Is Not Found In Oracle Internet Directory

A user connects to Forms and 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 OracleAS 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 a Oracle Internet Directory/DAS page 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 is customizable 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.

6.2.2 Support for Default Preferences in Oracle Internet Directory to Define Forms Resources

In previous releases, Forms uses resources added to each individual user account using the Oracle Delegated Administration Services. This implementation means that even if users share a common resource, it needs to be implemented for each user, no matter if there are 10 of them or 10,000.

In this Forms release, Forms and application administrators can define common used resources as default resources using the Oracle Internet Directory preferences. An

administrator creates a resource once and all user accounts automatically inherit this resource to be used within Forms.

6.2.3 Support for Dynamic Directives With Forms and OracleAS Single Sign-On

Enforcing OracleAS Single Sign-On in Forms is now done within the formsweb.cfg file. There is now a new OracleAS Single Sign-On parameter, `ssoMode`, to indicate when a custom application requires OracleAS Single Sign-On authentication.

This parameter allows a Forms Services instance to handle both application types, public and OracleAS Single Sign-On protected Forms. Because OracleAS Single Sign-On is configured in the formsweb.cfg file, Enterprise Manager Application Server Control Console can read and write the single OracleAS Single Sign-On parameter.

6.2.4 Support for Database Password Expiration for Forms Running with OracleAS Single Sign-On

In previous releases of Oracle Forms, password changes between Oracle Forms and an Oracle database would be successful, but these changes (including expirations) would not propagate to Oracle Internet Directory.

Now in OracleAS Forms Services, if the database password has expired and the *Forms Services* application, running in OracleAS Single Sign-On mode, is used to renew it, then 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 OracleAS Single Sign-On with Forms continues working even when a database password was changed. However, if password changes are made in SQL*PLUS, and not in Oracle Forms, then the database connect string is not updated in Oracle Internet Directory.

6.3 OracleAS Single Sign-On Components Used By Oracle Forms

The following software components in OracleAS are involved when running Forms applications in OracleAS Single Sign-On mode:

- Oracle Application Server Single Sign-On Server - an authentication Service in Oracle Application Server 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 Oracle Application Server Single Sign-On server, rendering authentication transparent for Oracle Application Server applications. OracleAS Forms Services and OracleAS Reports Services use `mod_osso` to register as a partner application to the Oracle Application Server 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 OracleAS 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, 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 OracleAS Single Sign-On. The formsweb.cfg file is located in the forms/server directory of an Oracle Application Server installation.

6.4 Enabling OracleAS Single Sign-On for an Application

Oracle Forms applications are configured using a central configuration file, the formsweb.cfg file in the forms/server directory. The formsweb.cfg file can be edited by using Enterprise Manager Application Server Control Console, which Oracle recommends.

OracleAS Single Sign-On and error handling are defined by the following parameters in the formsweb.cfg file:

- `ssoMode` [true | false]
- `ssoDynamicResourceCreate` [true | false]
- `ssoErrorUrl` [String URL]
- `ssoCancelUrl` [String URL]

These Oracle Forms parameters in the formsweb.cfg file can be set in the "User Parameter" section, to make them the default behavior for all Forms applications run by the server, and in a "Named Configuration", making the settings valid for a particular application only. A OracleAS Single Sign-On definition overrides the same definition set in the **User Parameter** section.

6.4.1 ssoMode

The `ssoMode` parameter enables an Forms Services application for OracleAS Single Sign-On. By default, Oracle Forms applications are not configured to run in OracleAS Single Sign-On mode. The `ssoMode` parameter can be set in two places in the formsweb.cfg file. Setting `ssoMode` as a system parameter with a value of `true` allows all applications to run in OracleAS Single Sign-On mode by this Forms Services instance. Setting the `ssoMode` parameter in a named configuration of an Oracle Forms application enables or disables OracleAS Single Sign-On only for this particular application:

```
[myApp]
form=myFmx
ssoMode=true
```

To enable OracleAS Single Sign-On for an application:

1. Start the Enterprise Manager Application Server Control Console.
2. Select **Forms**.
3. Select the **Configuration** tab.
4. Select the radio button next to the configuration section for your application and click **Edit**.
5. In the **Name** field, enter `ssoMode`.
6. In the **Value** field, enter `true`.
7. Click **Add New Parameter**.
8. Click Apply to update the formsweb.cfg file

   Single sign-on is now enabled for the selected application.

To disable OracleAS Single Sign-On for an application:

1. Start the Enterprise Manager Application Server Control Console.
2. Select **Forms**.
3. Select the **Configuration** tab.
4. Select the radio button next to the configuration section for your application and click **Edit**.
5. Select the radio button next to the `ssoMode` parameter.
6. In the **Value** column, enter `false`.
7. Click **Apply**.

   Single sign-on is now disabled for the selected application.

6.4.2 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 that displays is a standard form provided by the Oracle Delegated Administration Services. This Web page is not customizable 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 configuring an application as OracleAS Single Sign-On enabled with the value of the `ssoDynamicResourceCreate` parameter set to `false`, while not specifying a value for the `ssoErrorURL`, will cause 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'll need to set these parameters:

```
[myApp]
form=myFmx
ssoMode=true
ssoDynamicResourceCreate=false
```

For information about setting these parameters through Enterprise Manager Application Server Control Console, see Chapter 4.3.3, "Managing Parameters".

6.4.3 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 as a system parameter 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://myServ.com:7779/servlet/handleCustomRADcreation.jsp
…
```

6.4.4 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 he presses the cancel button in the HTML form that is used to dynamically create the RAD entry for the requested application.

6.4.5 Accessing Single Sign-on Information From Forms

Optionally, if you need to work with OracleAS Single Sign-On authentication information in a Forms application, the GET_APPLICATION_PROPERTY() built-in can be used to retrieve the following OracleAS Single Sign-On login information: OracleAS 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') ;
formsAppEntity := get_application_property('sso_formsid');
config := get_application_property('config').
```

Note: config can be obtained even in non-SSO mode

6.5 Integrating Oracle Forms and Reports

Oracle Reports is installed with OracleAS Single Sign-On 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 OracleAS Single Sign-On protected Oracle Forms application, the authenticated user's OracleAS Single Sign-On identity is implicitly passed to the Reports Server with each call to `RUN_REPORT_OBJECT` Built-in. The OracleAS Single Sign-On identity is used to authenticate the user to the Reports Server for further authorization checking, if required.

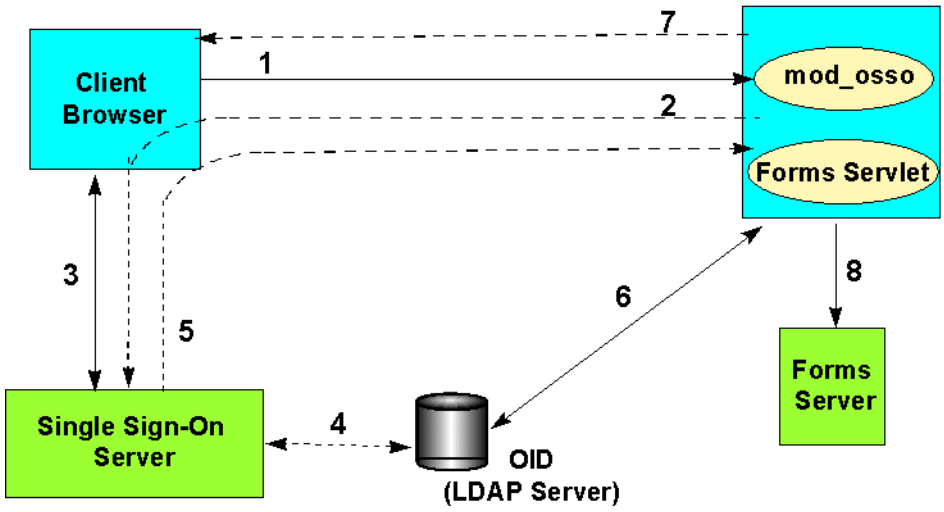
A Forms application running in non-OracleAS Single Sign-On mode can run a report on a Single Sign-on secured Reports Server, but fails if the Reports Server requires authorization. Also users must provide their OracleAS Single Sign-On credentials when retrieving the Reports output on the Web.

For more information about integrating Oracle Forms and Oracle Reports, see the white paper *Integrating Oracle Forms 10g and Oracle Reports 10g* at Oracle Technology Network http://www.oracle.com/technology/products/forms/.

6.6 Authentication Flow

The following is the authentication flow of OracleAS Single Sign-On support in Oracle Forms the first time the user requests an application URL that is protected by Oracle Application Server Single Sign-On:

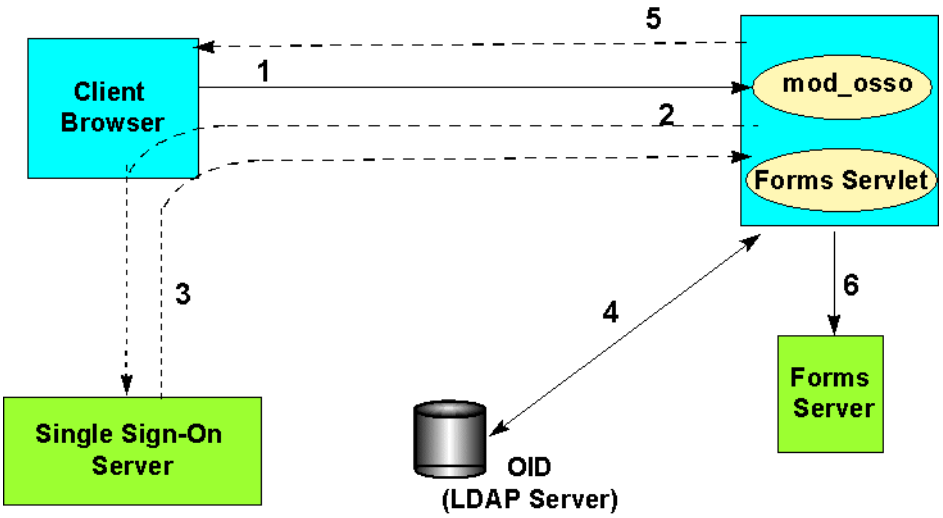
Figure 6–1 Authentication Flow for First Time Client Request



1. The user requests a Forms URL similar to `http(s)://<hostname>:<port>/forms/frmservlet?config=<application>&...`
2. The Forms Servlet redirects the user to the OracleAS Single Sign-On server.
3. The user provides user name and password through Login form.
4. The password is verified through Oracle Internet Directory (LDAP Server).
5. The user gets redirected to the URL with `sso_userid` information.
6. Forms Servlet gets the database credentials from Oracle Internet Directory.
7. Forms Servlet sets the user ID parameter in the Runform session and the applet connects to the Forms Listener Servlet.
8. Forms Servlet starts the Forms Server.

The following is the authentication flow of Oracle Application Server Single Sign-On support in OracleAS Forms Services when a user, authenticated through another Partner Application, requests an application that is protected by Oracle Application Server Single Sign-On.

Figure 6–2 Authentication Flow for Subsequent Client Requests



1. The user requests Forms URL.
2. Forms Servlet redirects the user to the OracleAS Single Sign-On Server.
3. The user gets redirected to the URL with `sso_userid` information.
4. Forms Servlet gets the database credentials from Oracle Internet Directory.
5. Forms Servlet sets the user ID parameter in the Runform session and the applet connects to the Forms Listener Servlet.
6. Forms Servlet starts the Forms Server.

7

JVM Pooling

This chapter contains the following sections:

7.1 Overview

JVM pooling allows administrators to consolidate the number of JVMs that are used so that the Forms sessions can share JVMs, rather than each one having its own instance. JVM pooling results in a large reduction of memory consumption, thus freeing up more resources on your server.

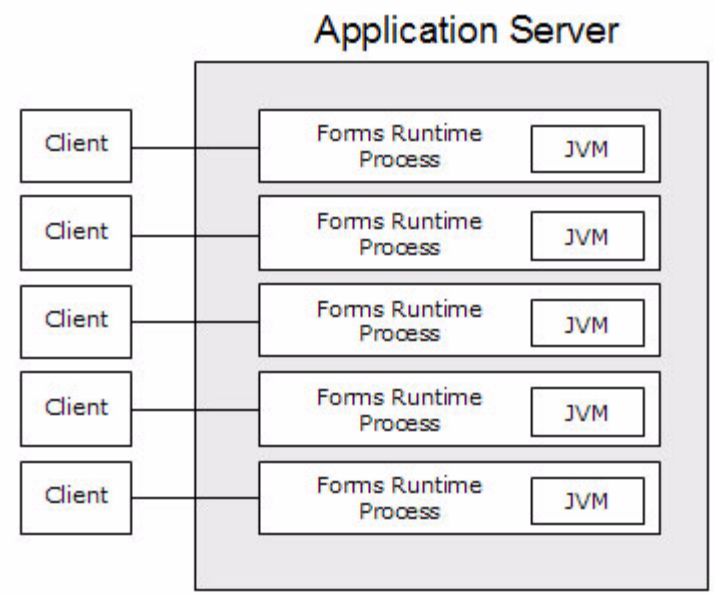
JVM pooling works in conjunction with the Java Importer. The Java Importer allows developers at design-time to reference Java classes from PL/SQL within the Forms Builder. At runtime, Forms uses a Java Virtual Machine (JVM) to execute Java code. JVM pooling expands the capabilities of a Forms application because it can now do operations that can't be done natively in Oracle Forms. For more information on the Java Importer, see the Oracle Forms Developer online help.

7.2 JVM Pooling Examples

Take for example an Oracle Forms application that has a user interface button. When the button is pressed, 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.

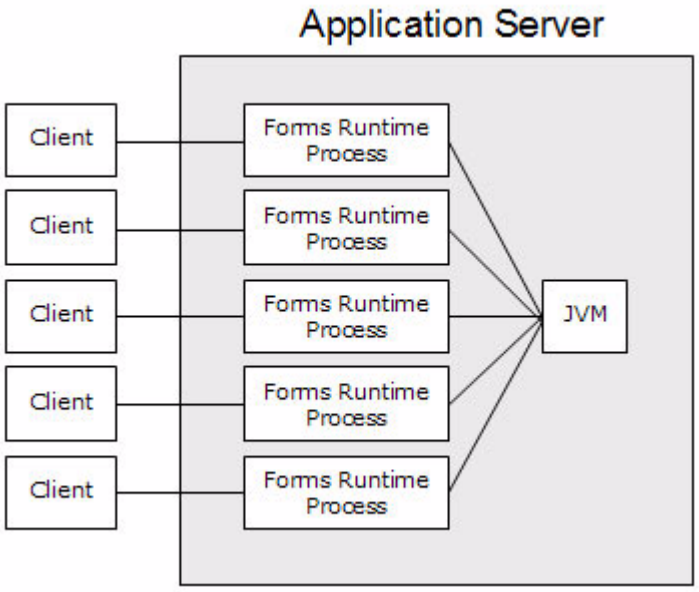
When JVM pooling is not enabled, the Forms Runtime Process operates as described in Figure 7–1, where each Oracle Forms session has its own instance of the JVM which is called an **in-process JVM**.

Figure 7–1 Forms Runtime with no JVM Pooling



When JVM pooling is enabled, ideally, one JVM can be used for multiple Forms processes, as described in Figure 7–2:

Figure 7–2 Forms Runtime with JVM Pooling Enabled



In this example, five JVM instances have been reduced to one, thereby reducing memory usage.

7.3 Design-time Considerations

This section contains the following:

7.3.1 About Previous Versions of the Java Importer

When the Java Importer was added to Oracle Forms, each Forms session that used the Java Importer had its own instance of the JVM to execute Java code. Each JVM consumes memory on the server, and if there are many concurrent users, the amount of memory consumed by the multiple JVM processes could become significant.

7.3.2 Re-importing Your Java Code

If you used the Java Importer feature of Oracle Forms prior to 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 needed by the Java Importer are different.

From Oracle Application Server Forms Services 10*g* and onwards, the Java Importer generates the "new" PL/SQL wrappers. If you want to use the Java Importer, but don't 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.

If you use Java in your application, but don't wish to use JVM pooling, then you do not need to re-import your Java classes. The in-process JVM will work with previously generated PL/SQL wrappers and with newly generated PL/SQL wrappers.

Note: With an in-process JVM, the JVM is part of the Oracle Forms Runtime Process itself, and is not a separate process. When JVM Pooling is used, the JVM is a separate, external process to the Forms Runtime Processes.

7.3.3 About Sharing Static Variables Across Multiple JVMs

If you used the Java Importer feature prior to JVM pooling, each Forms Runtime Process had its own in-process JVM that is separate from all of the others. With JVM pooling, you have multiple Forms runtime processes running Java code within the same JVM. One advantage 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'll 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 7.5.14, "About Child JVMs".

Prior to JVM pooling, if you changed interestRate it wouldn't affect any other instances because each Oracle Forms Services 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`.

7.4 About The JVM Controller

For each Oracle Forms session, there is one Oracle Forms Runtime Process on the application server. This process is where Oracle Forms actually runs, as well as manages the database connection; queries and updates data; runs any PL/SQL in the Form; executes triggers; and so on.

The Oracle Forms Runtime Process also contains the JVM to run Java in your application. As an optimization feature, the JVM is only started if the Oracle Forms application uses the Java Importer. For applications that do not use it, there is no extra memory consumption for the JVM.

JVM pooling is a new process that contains the JVM controller. With JVM pooling, the JVM runs outside of the Oracle Forms Runtime Process. The JVM can also be shared by multiple Oracle 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 Oracle Forms Runtime Process contains an in-process JVM.

Note: Use of JVM pooling is optional. Administrators can choose to not use JVM pooling and have the JVM contained in the Oracle Forms Runtime Process.

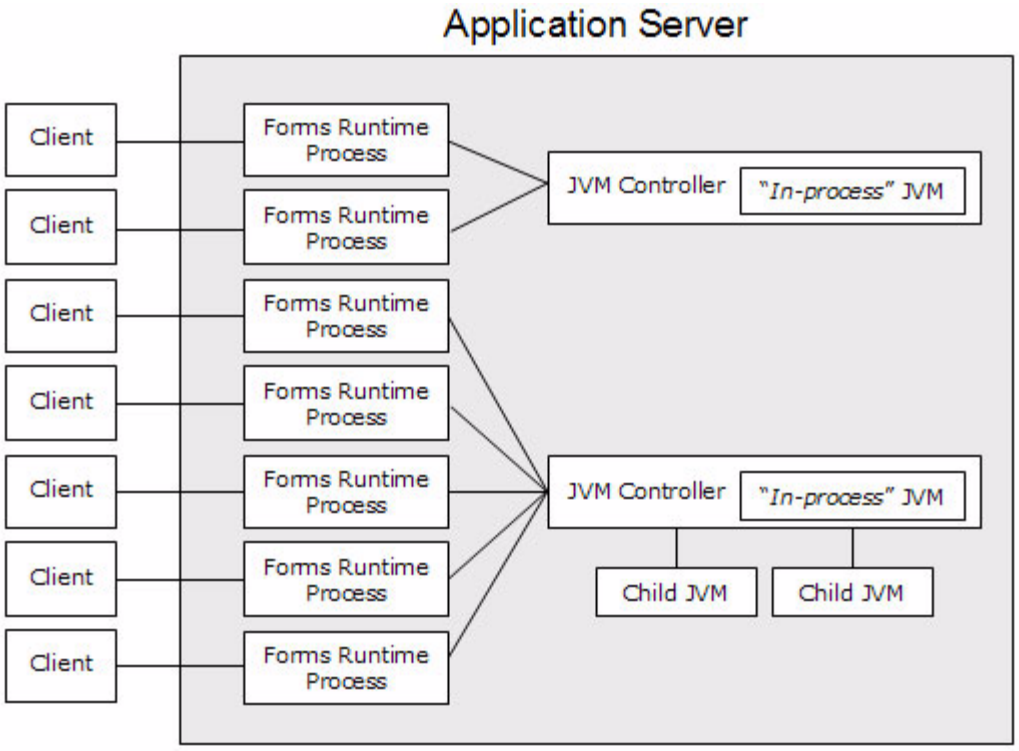
When an Oracle Forms Runtime Process needs to execute Java, it sends a message to the JVM that is contained in the JVM controller. The JVM creates a new thread for that Oracle Forms Runtime Process. The JVM then continues to listen for the next new request from a different Oracle Forms Runtime Process while the newly created thread processes the request and sends the results back to the Oracle Forms Runtime Process. For the life of this Oracle Forms session, the Oracle Forms Runtime Process communicates directly with that thread.

Since each Oracle 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 Oracle Forms applications may want to use different JVMs with different options or classpaths. You can specify which JVM controller an Oracle Forms application should use in the named sections of the Oracle Forms configuration file (formsweb.cfg). See Section 7.5.14, "About Child JVMs" for more information.

Figure 7–3 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 7–3 Multiple JVM Controllers with Child Processes



Although it's not shown in Figure 7–3, each JVM controller has a unique name which is used in starting and stopping, or for referencing in the Oracle Forms configuration file.

Figure 7–3 is conceptual only in that it shows different Oracle Forms applications using different JVM controllers. However, the Oracle 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.

7.5 JVM Pooling Management

While the Oracle Forms Runtime Process interacts directly with the JVMs, the JVM controller takes commands from an administrator, such as stopping the JVMs, or enabling or disabling logging, etc. For example, when an administrator issues a stop command, the command ensures all child JVMs are terminated.

The JVM controller can be managed in two ways:

- From Enterprise Manager Application Server Control (the recommended method by Oracle)
- From the command line

7.5.1 About Managing JVM Controllers from Enterprise Manager Application Server Control

Enterprise Manager Application Server Control provides a Web-based environment to manage all available JVM pooling options. Enterprise Manager Application Server Control interacts with the JVM controller so that administrators can manage JVM pooling. It is the central place for internally managing all of the JVMs for a JVM

controller. 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, Enterprise Manager Application Server Control also provides metric information such as resources (memory and CPU) that are consumed by JVM controllers.

Use the JVM page in Application Server Control to manage JVM pooling tasks:

- Section 7.5.3, "Creating a New JVM Controller"
- Section 7.5.5, "Editing JVM Controller Properties with Enterprise Manager Application Server Control"
- Section 7.5.6, "Specifying Default JVM Controller Properties"
- Section 7.5.4, "Deleting a JVM Controller"
- Section 7.5.7.1, "Starting or Restarting a JVM Controller"
- Section 7.5.8.3, "Stopping a JVM Controller"
- Section 7.6, "JVM Controller Logging Management"

The information that is displayed on the JVM Overview page includes:

Table 7–1 Application Server Control Overview Information for JVM Pooling

Item	Description
Total Memory Usage (%)	Shows memory usage statistics for private, shared, and total usage. Example: Private 0.8519 Shared 3.4226 Total 4.2745
Show All Details	When expanded, the following information displays for all JVM instances: Classpath
JVM Options Displays options that have been enable for this JVM	Log Directory Displays complete path to log file.
Comment	Start Time, example: Fri Aug 20 03:58:57 2004 PDT.
Hide All Details	Collapses all displayed information for all JVM instances.
Select	Use this radio button to select the target JVM you want to manage.
Details	Click Show (+) to expand or Hide (-) to collapse the selected JVM instance information.
Name	Displays the name of this JVM controller when it was created.
Status	Indicates whether the JVM controller is running or not.
Comment	Displays comments about the JVM controller.
CPU Usage (%)	Displays statistics for CPU usage as a percentage.
Private Memory Usage (%)	Displays private memory usage as a percentage.
JVMs	Displays the number of running instances of the target JVM.
Current Sessions	Displays the number of sessions attached to the target JVM.
Maximum Sessions per JVM	Displays the specified limit of number of sessions that can attach to a JVM.
Logging	Indicates whether or not logging is enabled for this JVM.
Log File	When logging is enabled, this icon provides a link to the log file for viewing.

7.5.2 About 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 Enterprise Manager Application Server Control with its user-friendly screens and online help.

Enterprise Manager Application Server 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.

7.5.3 Creating a New JVM Controller

After you've created a new JVM controller, you'll need to start it, as described in Section 7.5.7.1, "Starting or Restarting a JVM Controller".

To create a new JVM controller:

1. Click Create JVM Controller.

   The Create New JVM Controller page appears.

2. Enter the following information for the new JVM, as described in Table 7–2, " Options for Creating a New JVM Controller":

Table 7–2 Options for Creating a New JVM Controller

Option	Description
Name	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 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.
Maximum Sessions per JVM	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.
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.
JVM Options	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 parameters.
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.
Logging	On, Off, or Default. Setting to On or Off overrides the default JVM logging behavior. Setting to Default tells the new JVM to inherit default logging properties from the default JVM.
Comment	Add any comments about this JVM in this text area.

3. Click **Apply** to create the JVM with these settings.

   The Forms JVM Controllers page reappears.

4. Restart the JVM. See Section 7.5.7.1, "Starting or Restarting a JVM Controller" for more information.

7.5.4 Deleting a JVM Controller

Oracle recommends stopping a JVM controller before deleting it. If you delete it without stopping it, the JVM will not be removed from the JVM controllers page.

To delete a JVM controller:

1. Click the radio button to the left of the target JVM controller.
2. Click **Edit**.

   The Edit JVM Controller page appears.

3. Click **Delete**.

   The Confirmation page appears.

4. Click **Yes** to delete it.

   The Forms JVM Controllers page reappears without the deleted JVM controller.

7.5.5 Editing JVM Controller Properties with Enterprise Manager Application Server Control

You edit the properties for a JVM controller with Enterprise Manager Application Server Control, which provides an HTML-based graphical user interface.

To edit JVM controller properties:

1. Click the radio button to the left of the target JVM controller.
2. Click **Edit**.

   The Edit JVM Controller page appears.

3. Edit the following information for the new JVM, as described in Table 7–3, " JVM Controller Property Settings":

Table 7–3 JVM Controller Property Settings

Setting	Description
Maximum Sessions per JVM	Specifies the maximum number of concurrent Oracle Forms sessions this JVM will serve before a new JVM is spawned. This value will override any other that is set for the default JVM controller.
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.
JVM Options	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 parameters.
Log Directory	Leave Log Directory blank to use the default log location for the default JVM controller. If any other directory is set, the log file cannot be viewed through Enterprise Manager.

Table 7–3 (Cont.) JVM Controller Property Settings

Setting	Description
Logging	On, Off, or Default. Setting to On or Off overrides the default JVM logging behavior. Setting to Default tells the new JVM to inherit default logging properties from the default JVM.
Comment	Add any comments about this JVM in this text area.

4. Click **Apply** to commit the JVM property settings.

   The Forms JVM Controllers page reappears.

7.5.6 Specifying Default JVM Controller Properties

You can use the default JVM controller as a way for new JVM controllers to inherit predefined properties.

To edit the default JVM controller properties:

1. Click the radio button to the left of the default JVM controller.

   The Edit JVM Controller page appears.

2. Edit the following information for the default JVM:

Table 7–4 Default JVM Controller Options

Option	Description
Maximum Sessions per JVM	Specifies the maximum number of concurrent Oracle Forms sessions the default JVM will serve before a new JVM is spawned.
Classpath	When you specify a classpath, it will override the system classpath or any classpath specified in your environment.
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.
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.
Logging	On or Off.
Comment	Add any comments about this default JVM in this text area.

3. Click **Apply** to change the default JVM property settings.

   The Forms JVM Controllers page reappears.

7.5.7 Starting and Stopping JVM Controllers with Enterprise Manager Application Server Control

Enterprise Manager Application Server Control is the recommend tool for managing Oracle Forms Services, such as starting, stopping, and restarting a JVM controller.

7.5.7.1 Starting or 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. Enterprise Manager Application Server Control does this step for you.

To start a JVM controller that is not running:

1. Click the radio button to the left of the target JVM controller.
2. Click **Start**.

   When the JVM controller has started, you will see a confirmation note at the top of the Forms JVM Controllers page.

To restart a running JVM controller:

1. Click the radio button to the left of the target JVM controller.
2. Click **Restart**.
3. Click **Yes** on the Confirmation page.

   The Forms JVM Controller page reappears

   When the JVM controller has restarted, you will see a confirmation note at the top of the Forms JVM Controllers page.

To Stop a JVM Controller

1. Click the radio button to the left of the target JVM controller.
2. Click **Stop**.
3. Click **Yes** on the Confirmation page.

   When the JVM controller has been stopped, you will see a confirmation note at the top of the Forms JVM Controllers page.

7.5.8 JVM Controller Usage Commands

The JVM controller takes a command and various options. You must supply the name of the JVM controller as an argument for the JVM controller you want to manage.

The executable name for the JVM controller is `dejvm`. It is used to start and stop the JVM controller, as well as manage what it does.

The format of the command line is:

```
dejvm -<command> jvmcontroller=<controllerName> [options]
```

where:

- *command* is the command you are issuing to JVM controller.
- *controllerName* is the name of the JVM controller you are referring to.
- *options* is zero or more options available for the command you have chosen. See

  See Section 7.5.10, "JVM Controller Command Examples" for the list of available commands and options.

7.5.8.1 Command Restrictions

Keep these command restriction 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 below. If you are using Enterprise Manager Application Server Control, there are screens that have an interface for issuing these commands.

7.5.8.2 Starting a JVM Controller at the Command Line

Use the `-start` command and the following parameters to start a JVM controller, as described in Table 7–5, " Start Command Parameters":

Table 7–5 Start Command Parameters

Parameter	Description
`jvmcontroller`	Refers to the name of the JVM controller you wish to issue the command. This is also how the Forms Runtime Process identifies the JVM controller to send its requests to. It must be unique within a computer, but another JVM controller on a different computer may use the same name. The format of this parameter has the same restrictions as a filename. For instance, it cannot contain special characters such as `"/", ":", "*"`, etc. This parameter is required.
`maxsessions`	The maximum number of Forms runtime processes that a JVM can service before creating a child JVM. If `maxsessions` is exceeded, the JVM controller will automatically spawn a new child JVM process with the same settings as the JVM controller. When maxsessions is 65535, this means unlimited connections -- no child JVM will ever be spawned. This is useful if you discover through experience or research that a JVM can only handle a certain number of Forms runtime processes before performance of the JVM degrades. This parameter is optional. Default is 65535.
`logdir`	Location for the log file. The log filename will be automatically generated and will be `<jvm controller>.log`, where <jvm controller> is the name of the JVM controller. If you override this value, then you will not be able to view the log from Enterprise Manager. If you use the default log location (recommended), then you can view the log file from Enterprise Manager. This parameter is optional. The default is `ORACLE_HOME/tools/jvm/log`.
`classpath`	Classpath of the JVM. If you specify the classpath, the system classpath will be ignored, and only the classpath you specified will be used. This parameter is optional. The default is the system classpath or the classpath of the current environment.
`jvmoptions`	JVM options to specify. Refer to the Sun Java documentation for a list of valid JVM startup parameters. This parameter is optional. There is no default value. When specifying this parameter on the command line, use quotes around the value if it contains spaces. When specifying this value in the jvmcontrollers.cfg, do not use quotes, even if the value contains spaces.
logging	Specifies logging as ON or OFF. Default is ON.

7.5.8.3 Stopping a JVM Controller

Use the `-stop` command to stop the JVM controller. You must supply the name of the JVM controller as an argument for the JVM controller you want to stop. You will receive an error if a JVM controller with the specified name is not running. There is no additional option. See Section 7.5.10, "JVM Controller Command Examples" for more information.

7.5.9 The JVM Controller Configuration File

The JVM controller configuration file is used by Enterprise Manager and may optionally be used as a convenience for administrators at the command line. The name and location of the configuration file is:

```
ORACLE_HOME/tools/jvm/jvmcontrollers.cfg
```

> **Note:** You cannot change the location or name of the JVM controllers configuration file.

It works similarly 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.

When you start a JVM controller, it can take its settings from the configuration file, rather than having to be entered on the command line. You may specify none, some, or all options in this file, both in the default section and in named sections.

An example jvmcontrollers.cfg file might look like this:

```
#
# This is the default section. These parameters will
# apply unless overridden in a named section (lower down)
# or on the command line.
#
[default]
jvmoptions=-Xms512m -Xmx1024m
maxsessions=50

#
# Below are the named sections.
#
[hrJVM]
jvmoptions=-Xms256m -Xmx512m
classpath=/myJava/hrClasses
```

> **Note:** It's only when the `-start` command is used that the JVM controller uses the jvmcontrollers.cfg file. For all other commands, the jvmcontrollers.cfg file is not used.

7.5.9.1 Priority of Startup Options

This section describes the priority of how the startup options are applied. When you start a JVM, you must specify the `jvmcontroller` parameter. The JVM controller then follows these steps:

1. The JVM controller looks in the default section of jvmcontrollers.cfg and applies any options that are specified there.

2. The JVM controller looks in `jvmcontrollers.cfg` to see if there is a named section that corresponds to the jvmcontroller parameter. If so, it will take any options that are specified, overriding any it may have found in step 1.
3. The JVM controller then examines the command line arguments. Any options specified there override the options from steps 1 and 2.

This means that the command line parameters have the highest priority, followed by named sections in the JVM controller configuration file, followed by the default section, followed by default values or system settings (e.g. classpath).

For any commands not specified in the above steps, they will take their default values.

7.5.10 JVM Controller Command Examples

Here are some command line examples. It is assumed that the jvmcontrollers.cfg file is similar to the previous 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` will be `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 will be 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 will be `"-Xms128m -Xmx256m"` because the command line overrides the jvmcontrollers.cfg file. Similarly, `maxsessions` will be `75`. All other parameters take on their default values.

- `dejvm -start jvmcontroller=myJVM maxsessions=100 classpath=/myJava/myClasses;/moreJava/moreClasses`

  The controller will have `jvmoptions="-Xms512m -Xmx1024m"` as defined in the default section of jvmcontrollers.cfg. maxsessions will be 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.

7.5.11 Forms Configuration File Settings

This section describes the JVM pooling parameters that are used in the Forms configuration file (formsweb.cfg). The parameter names are not case-sensitive. Remember, you can use Enterprise Manager to administer the Forms configuration file. Table 7–6, " Oracle Forms JVM Controller Startup Parameters" describes the startup options that you can place in the formsweb.cfg file:

Table 7–6 Oracle Forms JVM Controller Startup Parameters

Parameter	Description
`jvmcontroller`	Valid values: See Section 7.5.8.2, "Starting a JVM Controller at the Command Line". In addition, you can specify no JVM. Default value: none 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 (see section 2.3, table 1). If jvmcontroller does not have a value, then the Oracle Forms Runtime Process will start its own in-process JVM, which means that the Java Importer uses pre-10*g* behavior.

7.5.12 Startup Example

The following is a snippet from a formsweb.cfg file the shows the startup flow:

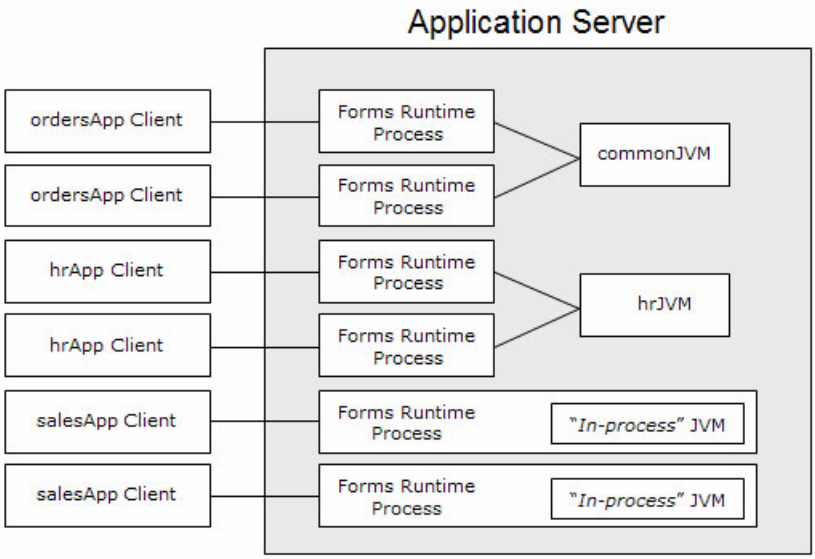
```
# System settings
[default]
jvmcontroller=commonJVM

[ordersApp]
form=orders.fmx
userid=orders/orderspw@orcl
[hrApp]
form=hr.fmx
userid=hr/hrpw@orcl
jvmcontroller=hrJVM
[salesApp]
form=sales.fmx
userid=sales/salespw@orcl
```

If a user starts an `ordersApp` application, and the application executes Java code, the Oracle Forms Runtime Process will route the request to the JVM controller named `commonJVM`. Because the `[ordersApp]` application section doesn't specify which JVM controller to use, the Oracle Forms Runtime Process uses the global one. If the JVM controller isn't 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 Oracle 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 isn't 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 Oracle 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 7–4:

Figure 7–4 Multiple JVMs for multiple applications



In Figure 7–4, the `commomJVM` controller, its in-process JVM, and any child JVM is represented as a single box, as well as the `hrJVMcontroller`.

7.5.13 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 Enterprise Manager. 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 don't want different applications to share static data.

7.5.14 About Child JVMs

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 Oracle 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 will not take any request from new Oracle Forms Runtime Processes. When a new Oracle Forms Runtime Process first attempts to execute Java code, it will attach to a JVM that is available, i.e. has fewer

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, then 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, then ordersJVM and its child JVMs do not affect – nor are not affected by – hrJVM or its child JVMs.

7.5.14.1 Child JVM Example

Suppose the JVM controller called ordersJVM has maxsessions=50. Each Orders application that is running sends requests to ordersJVM. Each time a new Oracle Forms Runtime Process sends a request to ordersJVM, a new thread is created that communicates with the Oracle 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 Oracle 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 not connect to hrJVM, only to child JVMs of ordersJVM.

7.6 JVM Controller Logging Management

When logging is enabled, the JVM controller logs certain information to the logfile:

- The values of the JVM parameters (maxsessions, classpath, etc.);
- When a JVM controller starts and stops;
- When a child JVM is spawned;
- When an Oracle Forms Runtime Process starts a new connection, along with its process ID

  This is useful for knowing which Oracle Forms Runtime Processes are connected to which JVM controller for diagnostics or administration;
- When an Oracle Forms Runtime Process session ends and disconnects from the JVM.

7.6.1 Enabling and Disabling Logging

This section contains the following:

7.6.1.1 Specifying Default Logging Properties

1. Select the default JVM controller.
2. Click **Edit**.
3. The Edit JVM Controller page appears.
4. From the logging dropdown box, select On or **Off**.
5. Click **Apply**.

   The Forms JVM Controllers page reappears.

7.6.1.2 Specifiying the Log File Directory Location

You can specify the log file directory when you create or edit a 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 when creating or editing a JVM controller:

1. Click **Create JVM Controller** when creating a new controller, or select the radio button next to the target JVM controller.

   The Create New JVM Controller page appears or the Edit JVM Controller page appears.
2. Enter the following information for the JVM log file location:

   Log Directory

   Leave **Log Directory** empty to use the default log location for the default JVM controller. If any other directory is set, the log file cannot be viewed through Enterprise Manager.
3. Click **Apply** to create the JVM with these settings. The new JVM is automatically started.

   The Forms JVM Controllers page reappears.

7.6.1.3 Accessing Log Files

When logging is enabled for a JVM controller, a clickable icon in the Log File column will appear. When logging is disabled, the log file entry for that JVM controller is empty. See Section 7.6.1, "Enabling and Disabling Logging" for more information.

To access a log file:

1. Click the Log File icon in the Log File column that is available for that JVM controller.

   The Log File page appears and provides the following information:

Table 7–7 Log File Page Information

Item	Description
Log File	Contains the name of the selected JVM controller
Component Name	Displays the name of the JVM controller
Component Type	Displays the application type
Modified	Displays the last time the log file was updated
Size	Displays the log file size in bytes
Log File Contents	Table that contains the most recent log entries. A maximum of 2000 lines is retrieved.
Log Text	Displays the latest entries in the log file

7.6.1.4 Deleting a Log File for a JVM Controller

To delete a log file for a JVM controller:

1. From the JVM Controllers page, select the radio button next to the target JVM.
2. Click **Delete Logfile**.

   The Delete Confirmation page appears.
3. Click **Yes**.

   The JVM Controllers page reappears.

7.7 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.

8

Tracing and Diagnostics

This chapter contains the following sections:

8.1 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 Application Server Control Console, 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.

8.2 Configuring Forms Trace

An event is something that happens inside Oracle Forms as a direct or indirect result of a user action. 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 **Forms Trace Configuration** selection in the **Configuration** tab of Oracle Enterprise Manager 10*g* Application Server Control Console Forms 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 formsweb.cfg and default.env files before editing them with Oracle Enterprise Manager 10*g* Application Server Control Console.
- As with most Web applications, it is easy to lose unsaved changes by switching pages. Be sure to save any changes you make through Oracle Enterprise Manager 10*g* Application Server Control Console 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 the deletion of a single entry.
- If you manually edit any of the configuration or environment files, you'll need to restart Enterprise Manager as well as restart all Distributed Configuration Management (DCM) processes so that Enterprise Manager can read all changes. If you do not restart Enterprise Manager as well as DCM processes, any changes that you make through Enterprise Manager will overwrite any manual changes you've made to these files. These DCM processes include:
  - `emctl stop agent`
  - `emctl stop em`
  - `dcmctl stop`
  - `opmnctl stopall`
  - `opmnctl startall`
  - `dcmctl start`
  - `emctl start agent`
  - `emctl start em`

See Section 8.5, "List of Traceable Events" for a list of events and their corresponding event numbers.

To configure Forms Trace:

1. Start the Enterprise Manager Application Server Control Console.
2. From the Enterprise Manager Application Server Control Console main page, select the link to the Forms Services instance that you want to configure.
3. From the Overview page for the Forms Services instance, select the **Configuration** link.

To create a new parameter in the ftrace.cfg file:

- Enter a **Name** and **Value** for this new parameter and click **Add New Parameter** at the bottom of the page.

To delete a parameter in the ftrace.cfg file:

- Click the radio button next to the parameter to be deleted, then click **Delete**. Confirm the deletion on the next page.

To edit an existing parameter in the ftrace.cfg file:

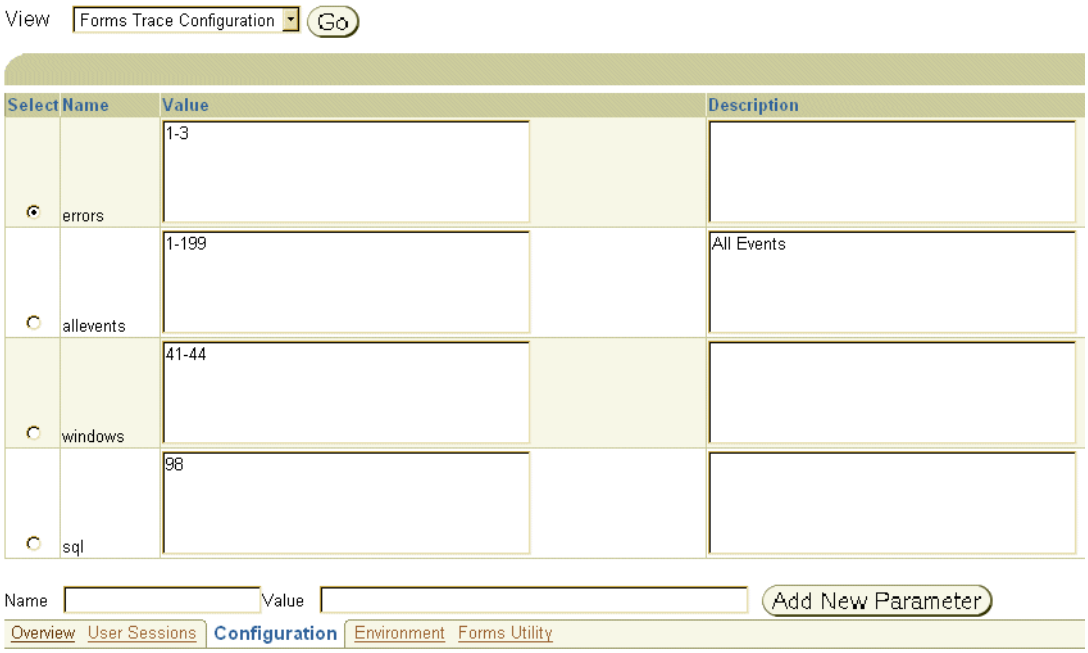
- Select the radio button next to it, and modify the values in the text areas. Click **Apply** to save your changes.

To save your changes:

- Click the radio button next to a parameter, then click **Apply**.

Figure 8–1 is a sample ftrace.cfg configuration file where three event sets have been specified.

Figure 8–1 Configuring Trace Events in Enterprise Manager



Note the following if you are manually editing ftrace.cfg:

- There must be a blank line between keyword entries.
- An Event group can have any name as long as they do not contain spaces. For example, a_b_c is an acceptable keyword.
- There must be a comma between each event number.
- You can use a range of numbers

When you start the trace, you can specify tracegroup = "custom1" on the command line, which is equivalent to specifying tracegroup = "32-46, 65, 66, 96, 194"

8.2.1 Specifying URL Parameter Options

The following command line parameters are used to configure Forms Trace:

```
Tracegroup =
Log = <filename>
```

Table 8–1, " Forms Trace Command Line Parameters" describes the parameter values:

Table 8–1 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 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 SQLInfo.
		▪ 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`

`Log`	Directory	Specifies where trace information is saved. Trace files must be saved to ORACLE_HOME/forms/trace for Enterprise Manager to find and process them correctly. If a directory is not specified, the file is written to the current working directory. If a log file is not specified, the process ID (PID) of the user process is used as the name of the trace file, for example, forms_<pid>.trc.

8.3 Starting Forms Trace

You start a trace by specifying trace entries in the URL or from Enterprise Manager Application Server Control Console. Entries should include the grouping of events to collect and the trace file name. Trace collection starts when the form executes.

> **Note:** You'll need to provide the credentials in the dialog box that displays (the user name and password that is required is for the operating system account that was used when Forms Services was installed).

The following are sample URLs to start a trace:

```
http://cx-pc/forms/frmservlet?form=cxl&record=forms&tracegroup=0-199
http://cx-pc/forms/frmservlet?form=cxl&record=forms&tracegroup=mysql
http://cx-pc/forms/frmservlet?form=cxl&record=forms&tracegroup=0-199;log=run1.log
```

A later release of Oracle Forms will implement a method for starting a trace via a built-in. The most recent information regarding Oracle Forms, including updated documentation, whitepapers, and viewlet demonstrations, is available on OTN at http:/www.oracle.com/technology/products/forms/.

8.4 Viewing Forms Trace Output

Only users with administrator privilige can view trace log files. The user needs to log in as a user that is in the administrators' group. By default, user 'admin', which is present as a default user in the 'adminstrators' group, can be used. Users can then be created and added later into this group to for logging in and viewing trace log files.

Once the user has logged in, he will not have to log in again in the same browser session to view trace log files for different formsweb sessions.

Trace data is stored in a binary file with a *.trc extension. If you're not using Enterprise Manager Application Server Control Console, you'll need to use the Translate utility.

> **Note:** The parameter `allow_debug` must be set to `true` in the default section of the Forms Web Configuration file before trace logs can be viewed from the User Sessions screen in the Enterprise Manager Application Server Control Console.

To view trace data, use Enterprise Manager:

1. In Enterprise Manager Application Server Control Console, select the **User Sessions** link.
2. Click **View Trace Log** to see the contents of the trace log.

8.4.1 Running the Translate Utility

The Translate utility converts trace data to XML or HTML formats. You'll need to specify an additional parameter "OutputClass" which has two legal values: "WriteOut" and "WriteOutHTML". If you use "WriteOut", the output file will be in XML format. If you use "WriteOutHTML", the output file will in HTML format.

These two values ("WriteOut" and "WriteOutHTML") are case-sensitive.

To convert trace data to XML format:

- At the command line, enter:

```
java oracle.forms.diagnostics.Xlate datafile=a.trc
outputfile=myfile.xml outputclass=WriteOut
```

to create `myfile.xml`.

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
```

8.5 List of Traceable Events

Table 8–2, " List of Traceable Events" lists the events that can be defined for tracing. In future releases of Forms, more events will 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 creates a greater quantity of information about the event (for example, argument values).

Table 8–2 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	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

Table 8–2 (Cont.) List of Traceable Events

Event Number	Definition	Type
44	Window Resize	point
45	Tab Page	point
46	Timer	point
47	Reserved for future use	NA
48	Reserved for future use	NA
49-63	Reserved	NA
64	Form (Start & End)	duration
65	Procedure (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	NA
96	Builtin (Start & End)	builtin
97	User Exit (Start & End)	duration
98	SQL (Start & End)	duration
99	MenuCreate (Start & End)	duration
100	PLSQL (Start & End)	duration
101	Execute Query	duration
102-127	Reserved	NA
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	NA
160-191	Reserved	NA
192*	Environment Dump	N/A
193*	State Delta	N/A

Table 8–2 (Cont.) List of Traceable Events

Event Number	Definition	Type
194*	Builtin Arguments	N/A
195*	UserExit Arguments	N/A
196*	Procedure Arguments	N/A
197*	Function Arguments	N/A
256 and higher	User defined	NA
1024 an higher	Reserved for internal use	NA

* 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.

8.5.1 List of Event Details

The following tables list event details that can be defined for tracing:

- Table 8–3, " User Action Event Details"
- Table 8–4, " Forms Services Event Details"
- Table 8–5, " Detailed Events"
- Table 8–6, " Three-Tier Event Details"
- Table 8–7, " Miscellaneous Event Details"

8.5.1.1 User Action Events

Table 8–3 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
Window Activate, Deactivate,Close, Resize	WindowName, FormName, Size	41,42,43,44

8.5.1.2 Forms Services Events

Table 8–4 Forms Services Event Details

Event Name	Details	Number
Form	Form ID, Name, Path, Attached Libraries, Attached Menus	64
Procedure	Procedure 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

8.5.1.3 Detailed Events

Table 8–5 Detailed Events

Event Name	Details	Number
Builtin	BuiltinName, FormId	96
User Exit	UserExitName, FormId	97
MenuCreate	MenuName, FormID	99
PLSQL	PLSQLSTmt, FormID	100
ExecQuery	Block Name	101

8.5.1.4 Three-Tier Events

Table 8–6 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

8.5.1.5 Miscellaneous Events

Table 8–7 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
Function Args	Arguments (in | out) passed to a procedure	197

8.6 Monitoring Forms Services Trace Metrics

Use this Enterprise Manager page to review Forms Services Trace metrics.

1. Start the Enterprise Manager Application Server Control Console.
2. From the Enterprise Manager Application Server Control Console main page, select the link to the **User Sessions** link
3. Click the icon in the **View Trace Log** column.

You can write your own classes to implement a write.output class that is XML-based.

8.7 Servlet Logging Tools

Servlet logging tools enable site administrators to keep a record of all Oracle Forms sessions, monitor Oracle Forms-related network traffic, and debug site configuration problems.The features of servlet logging tools available with Oracle Application Server 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)

These sections on the servlet logging tools contain the following:

- Section 8.7.1, "Enabling Logging"
- Section 8.7.2, "Location of Log Files"
- Section 8.7.3, "Example Output for Each Level of Servlet Logging"

8.7.1 Enabling Logging

You enable logging by:

- Appending one of the strings in Table 8–8, " Supported logging capabilities" to the serverURL parameter in the URL that starts the form.

- Appending one of the strings in Table 8–8, " Supported logging capabilities" to the serverURL client parameter in the **Configuration** page of Enterprise Manager Application Server Control Console.

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 8.7.3, "Example Output for Each Level of Servlet Logging".

Table 8–8 Supported logging capabilities

String appended to serverURL client parameter	Description of logging
(none)	No log messages are produced. However, during Forms Servlet initialization, a message is written to the log file stating the name and path of the configuration file being used.
/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 very verbose and is intended mainly for debugging and support purposes.

8.7.1.1 Specifying Logging in the URL

As an example, to start a performance-level trace, you would start the Oracle Forms application using a URL as follows:

```
http://yourserver/forms/frmservlet?serverURL=/forms/lservlet/
perf
```

8.7.1.2 Specifying Logging through Enterprise Manager

As an example, to start session-level logging for all users, you would change the serverURL entry in the default section in the Forms Web Configuration page to the following:

```
serverURL=/forms/frmservlet/session
```

8.7.1.3 Specifying Full Diagnostics in the URL that Invokes the Forms Servlet

As an example, to start full diagnostics, you would start the Oracle Forms application using a URL as follows. Note that if you append /debug to the URL used to invoke the Forms Servlet that servlet will output debug messages to the log file too.

```
http://yourserver/forms/frmservlet/debug?serverURL=/forms/
lservlet/debug
```

8.7.2 Location of Log Files

The servlet log file is application.log. It is written to the application-deployments/formsapp directory of the OC4J instance to which Forms is deployed.

In Oracle Application Server Forms Services, the full path is:

```
ORACLE_HOME/j2ee/OC4J_BI_FORMS/application-deployments/formsapp/
OC4J_BI_Forms_default_island_1/application.log
```

In Forms Developer, it is:

```
ORACLE_HOME/j2ee/DevSuite/application-deployments/
forms/application.log
```

8.7.3 Example Output for Each Level of Servlet Logging

The following are examples of the type of output you will get when you use the following levels of logging:

- (none)
- /session
- /sessionperf
- /perf
- /debug

8.7.3.1 (none)

```
FormsServlet init():
configFileName:     d:\Oracle/forms/server/formsweb.cfg
testMode:           false
```

8.7.3.2 /session

Session start messages (example):

```
Forms session <10> started for test-pc.mycompany.com ( 138.56.98.72 )
Forms session <10> runtime process id = 373
```

Session end message (example):

```
Forms session <10> ended
```

8.7.3.3 /sessionperf

```
Forms session <3> started for test-pc.mycompany.com ( 138.56.98.72 )
Forms session <3> runtime process id = 460
Forms session <3> ended
  Total duration of network exchanges: 1.041
  Total number of network exchanges: 2 (1 "long" ones over 1.000 sec)
  Average time for one network exchange (excluding long ones): 0.030
Total bytes: sent 1,110, received 316
```

8.7.3.4 /perf

```
Forms session <3> started for test-pc.mycompany.com ( 138.56.98.72 )
Forms session <3> runtime process id = 460
Forms session <3>: request processed in 1.011 sec. Received 8 bytes,returned 8
bytes.
Forms session <3>: request processed in 0.030 sec. Received 308 bytes, returned
1,102 bytes.
Forms session <3> ended
  Total duration of network exchanges: 1.041
  Total number of network exchanges: 2 (1 "long" ones over 1.000 sec)
  Average time for one network exchange (excluding long ones): 0.030
  Total bytes: sent 1,110, received 316
```

8.7.3.5 /debug

Here is an example run by going to a URL like http://test-machine:8888/forms/frmservlet/debug&config=myapp&serverURL=/forms/lservlet/debug):

```
=============== FormsServlet ===============
GET request received, cmd=debug,
qstring=config=myapp&serverURL=/forms/lservlet/debug
No current servlet session
File baseie.htm not found, looking in d:\Oracle/forms/server
The SSO_USERID is: null
=============== FormsServlet ===============
GET request received, cmd=startsession, qstring=config=myapp&serverURL=
/forms/lservlet/debug&ifcmd=startsession
No current servlet session
New servlet session started
SSO_USERID in startSession: null
SSO_AuthType in startSession: null
User DN: null
Subscriber DN: null
EM mode in the config file: 0
File default.env not found, looking in d:\Oracle/forms/server
envFile = d:\Oracle\forms\server\default.env
serverURL: /forms/lservlet/debug
rewrittenURL: /forms/lservlet/debug;jsessionid=27f6412da05c
426ab47db4ae77636113
=============== ListenerServlet ===============
GET request received, cmd=getinfo,
qstring=ifcmd=getinfo&ifhost=test-pc.mycompany.com&ifip=130.35.96.71
Existing servlet session, id = 27f6412da05c426ab47db4ae77636113, not from cookie
Creating new Runtime Process using default executable
Starting Forms Server in EM mode
startProcess: executing frmweb server webfile=HTTP-0,0,1
Getting stdin, stdout and stderr of child process
Writing working directory to stdin: d:\Oracle\forms
New server process created
Forms session <4> started for test-pc.mycompany.com ( 138.56.98.72 )
**********************************************
Got POST request, length = 8
HTTP request headers:
      ACCEPT-LANGUAGE: en
      PRAGMA: 1
      CONTENT-TYPE: application/x-www-form-urlencoded
      ACCEPT: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
      USER-AGENT: Mozilla/4.0 (compatible; MSIE 5.0; Win32)
      HOST:test-machine:8888
      CONTENT-LENGTH: 8
      CONNECTION: Keep-Alive
Existing servlet session, id = 27f6412da05c426ab47db4ae77636113, not from cookie
Forms session <4> runtime process id = 474
Port number is 2791
RunformProcess.connect(): connected after 1 attempts
Connected to ifweb process at port 2791
Forms session <4>: request processed in 1.032 sec. Received 8 bytes,
returned 8 bytes.
**********************************************
```

9

Configuring End User Monitoring

This chapter contains the following sections:

9.1 About End User Monitoring

End User Monitoring is a utility that is part of Oracle Enterprise Manager. It allows developers and administrators to monitor the performance of their applications. Since there is minimal overhead in terms of system resources when reporting live data, a system administrator is able to accurately monitor the performance of live applications without needing to take any special additional steps.

End User Monitoring includes reports and performance charts, along with system generated alerts. Unlike existing monitoring mechanisms within Forms, End User Monitoring records the delay experienced by the user from their computer, and not just the processing time inside the Oracle Forms Runtime Process.

The data that is reported from End User Monitoring is qualitative. It can be used for trend analysis, reports comparing different domains or user groups. For example, if opening a particular form is normally reported as taking 2 seconds, and this suddenly drops to 10 seconds, then it is reasonable to expect a corresponding degradation as experienced by users. It does not mean that a user sitting with a stopwatch would record exactly 10 seconds for the operation. Similarly, if a particular query normally reports a database time of 3 seconds, and that query time suddenly jumps to 30 seconds, then a similar performance degradation will be experienced by users.

It's also important to realize that End User Monitoring is not a debugging tool. Analyzing the data will help identify what areas to investigate further, but it will not in itself identify the cause.

9.2 Configuring End User Monitoring

You'll need to read and work through these sections to configure Oracle Forms Services and End User Monitoring:

- Section 9.2.4, "Modifying the Default Minimum Hits Threshold"
- Section 9.2.5, "Modifying the Exclusion of All Unreasonable Response Times"
- Section 9.3.1, "Modifying formsweb.cfg"

9.2.1 Requirements for Using End User Monitoring

In order to use End User Monitoring with Oracle Forms Services, you'll need to install and configure:

- Oracle Application Server 10.1.2 that includes Oracle Forms Services
- Access to, or an installation of Oracle Enterprise Manager Grid Control

  You'll need to know the computer name, configured port, and password to an Enterprise Manager Grid Control instance
- Oracle Management Agent, part of Enterprise Manager Grid Control

  The Oracle Management Agent is a process that is deployed on each monitored host. The Oracle Management Agent is responsible for monitoring all targets on the host, for communicating that information to the middle-tier Management Service, and for managing and maintaining the host and its targets.

  You install the Oracle Management Agent on the Oracle Application Server middle-tier. During installation, you'll need to specify the computer that runs Enterprise Manager Grid Control along with its configured port number and password.

9.2.2 Configuring Web Cache to Use End User Monitoring

Web Cache is the intermediary between Oracle Application Server (which contains the Forms Runtime Process) and Oracle Management Agent. In these steps, you configure a specific Web Cache instance to use End User Monitoring.

To configure Web Cache to use End User Monitoring:

1. Open the Web Cache Admin page for Oracle Application Server (port 9400 by default).
2. Click **Web Cache Admin** and login as the Web Cache administrator.
3. Under **Logging and Diagnostics**, click **End-User Performance Monitoring**.
4. From the Cache-Specific End-User Performance Monitoring table, select the radio button next to the target Web Cache and click Enable.
5. From the **Site-Specific End-User Performance Monitoring** table, select the radio button next to the middle-tier that contains the Oracle Monitoring Agent, and click Enable.
6. Click **Access Logs** under **Logging and Diagnostics**.
7. Change `access_log` format style to **End-User Performance Monitoring Format**.
8. Click **Apply Changes** and restart Web Cache
9. Access the target Oracle Application Server in a Web browser, then open the browser's view of the source for the Web page.

   If you can see `<SCRIPT SRC="/oracle_smp_EndUserMonitoring/oracle_smp_EndUserMonitoring.js"></SCRIPT>` at the end of the HTML page, End User Management is successfully enabled.

In the next sections, you select this Web Cache instance to be monitored from Enterprise Manager Grid Control, then configure Forms Services to monitor applications.

9.2.3 Specifying a Web Cache Instance to Monitor with Enterprise Manager Grid Control

In this part of the configuration, you specify the Web Cache instance Enterprise Manager Grid Control is to monitor. This Web Cache instance must have End User Monitoring enabled.

To add a Web Cache Instance to monitor with Enterprise Manager Grid Control:

1. Open `https://em.computer.company.com/em`.
2. Log in with the user name and password.
3. Select **Targets | Web Application | Add**.
4. Enter the name and Forms URL for the home page URL, for example `http://computer.company.com/forms/frmservlet`, and click **Next**.
5. Select **Forms** from **Available Targets** and select the instance name, then click **Next**.
6. Select the agent port (e.g. 1831) and click **NextFinish** and then . It may take a few minutes for the status to appear.
7. Once the status appears, go to **Page Performance** and click **Configure Web Application Web Caches**.
8. In **Configure Web Application Web Caches**, click **Add or Remove Web Application Components**.
9. Choose **Web Cache** from **Available Targets** and select a Web Cache target and click **Next**, then click **Apply**.
10. Select **Collecting** and set the interval to 1 minute.
11. Click **Apply**.

9.2.4 Modifying the Default Minimum Hits Threshold

Changing the Default Minimum Hits Threshold setting can significantly speed up data refreshing based on a specified number of minimum hits. A lower number means that data refreshes more often when a specified number of hits has been reached. A higher number means that data will refresh when a specified higher number of hits has been reached.

To change the default minimum hits threshold:

- Run the SQL scripts against the Enterprise Manager database:

```
update mgmt_parameters set parameter_value = 1 where parameter_name = 'mgmt_rt_
min_hits';
commit;
```

9.2.5 Modifying the Exclusion of All Unreasonable Response Times

The default unreasonable threshold is set to 60,000 milliseconds, which may be too small for Oracle Forms Applications. You may want to change this default to 1 minute.

To change the exclusion of unreasonable response times:

- Run the SQL Scripts against the Enterprise Manager database:

```
update mgmt_parameters set parameter_value = 3600000 where parameter_name =
'mgmt_rt_max_elapsed_time';
commit;
```

9.3 Enabling End User Monitoring

After configuring End User Monitoring, you can enable it to monitor applications by creating a new section in formsweb.cfg.

9.3.1 Modifying formsweb.cfg

You should create a new section in formsweb.cfg to monitor specific applications. For more information on creating specific sections in formsweb.cfg, see Section 4.3.2, "Managing Configuration Sections".

1. Open `ORACLE_HOME/forms/server/formsweb.cfg`.
2. Set `EndUserMonitoringEnabled=true`
3. Set `EndUserMonitoringURL=http://computername:7777/oracle_smp_EndUserMonitoring/oracle_smp_EndUserMonitoring_sdk.gif`

> **Note:** The computer name is the middle tier installation where Web Cache is running.

9.4 Additional Sources of Information

You can obtain additional information about End User Monitoring and information about interpreting metrics from these sources:

- Enterprise Manager Online Help from any metrics page
- *Oracle Enterprise Manager Concepts* available on the Oracle Application Server Documentation CD.

10

Performance Tuning Considerations

This chapter contains the following sections:

Tuning the connection between Oracle Application Server Forms Services and the Oracle Database Server is beyond the scope of this chapter.

10.1 Built-in Optimization Features of Forms Services

The Oracle Application Server Forms Services and Java client include several optimizations that fit broadly into the following categories:

10.1.1 Monitoring Forms Services

Use Oracle Enterprise Manager 10*g* Application Server Control Console to monitor Oracle Application Server Forms Services and review metrics information, including:

- Forms Services Instances
- Events
- User Sessions
- Forms Trace

10.1.1.1 Monitoring Forms Services Instances

Use the Overview page to monitor metrics for a Forms Services instance.

1. Start Enterprise Manager Application Server Control Console.
2. From the Enterprise Manager Application Server Control Console main page, select the link to the Forms Services instance that you want to monitor.

The Overview page for the Forms Services instance displays the following: Current Forms Services instance status (up, down)

- URL of the Forms Services instance being monitored
- Oracle Home of the Forms Services instance being monitored
- Percent CPU usage for all forms runtime processes for this instance of Forms Services
- Percent memory usage for all forms runtime processes for this instance of Forms Services
- Number of users logged in
- Response time is the ping time from Forms Enterprise Management agent to the forms servlet when the Enterprise Manager page is loaded.

Additionally, you can jump to the following detail pages:

- Session Details
- Forms Services Configuration
- Environment
- Forms Trace Configuration
- Forms Utility

10.1.1.2 Monitoring Forms Events

Use the Enterprise Manager Application Server Control Console to enable tracing for all events or specific ones.

10.1.1.3 Monitoring Metrics for User Sessions

1. Start the Enterprise Manager Application Server Control Console.
2. From the Enterprise Manager Application Server Control Console main page, select the link to the Forms Services instance that you want to monitor.
3. From the Overview page for the Forms Services instance, select the User Sessions link.

   This page shows the following information about each user session for the Forms Services instance:

   - PID: The process ID of the user session.
   - CPU usage: The percent CPU used by the runtime process.
   - Memory usage: The percent memory used by the runtime process.
   - Client IP Address: The IP address of the client computer used to connect to Forms Services.
   - Database User Name: The database username used by the Forms application for the user session.
   - Time of connection: The time when the user connected to Forms Services.
   - Trace Status: Indicates if tracing is ON or OFF.
   - View Trace Log: Allows a user to view the trace log.
   - Configuration Section: Opens the Edit Section page for the configuration section used by a particular forms session.

10.1.1.4 Sorting Metric Information

You can sort (in ascending order) on Process ID, CPU, Memory Usage, IP, User Name and Connect Time by clicking the link in the column header.

10.1.1.5 Searching

Use Search to locate specific metric information.

To search for session details:

- Select **Username**, **IP Address** or **PID** from the pulldown, enter an exact, case sensitive match in the following field, and click **GO**.

To view the complete list of sessions again after a search:

- Click **GO**.

10.1.2 Forms Services Web Runtime Pooling

Forms Runtime Pooling 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 situataions 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.

10.1.2.1 Configuring Prestart Parameters

Use Enterprise Manager Application Server Control Console to configure runtime pooling for Forms Services with the following parameters:

Table 10–1 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 executables that should be spawned initially	1
`prestartTimeout`	integer	Time in minutes for which the pre started executables to exist	0 (When set to zero the timer never starts)
`prestartMin`	integer	Minimum number of runtime executables to exist in the pool.	0

Table 10–1 (Cont.) Forms Runtime Pooling Parameters

Parameter Name	Data type	Description	Default Value
`prestartIncrement`	integer	The number of runtime executables to be created when below the minRuntimes.	0

Note that `prestartMin` defines the minimum number of pre-started runtimes that must exist at any time 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 OC4J, the various values provided for the above parameters are on a per JVM basis, and not the total for the application.

10.1.2.2 Starting Runtime Pooling

An administrator has the capability to pre-start the specified number of executables for a particular application from the Enterprise Manager Application Server Control Console. The administrator selects the required application, which alerts Forms Services. The Forms Servlet will be loaded on the start of the Web server (OC4J).

During initialization of the Forms Servlet, the formsweb.cfg file is read and the server prestarts the applications which has the prestartRuntimes parameter enabled.

10.1.3 Forms Services Utilities

The Forms Utility page provides a simple user interface to call a set of operations on the middle tier. These features will be enhanced in future releases.

Presently, only `ps` (to obtain process information) and a number of arguments are available.

10.1.3.1 To use the Forms Services Utility:

- In the **Parameter** text field, type:

  `ps`
  then click **Submit**.

  A list of processes is returned in the status window below.

10.1.4 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.

10.1.5 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.

10.1.6 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 Application Server Forms Services communicates with the Java client using meta data messages. Meta data 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 Application Server 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 Application Server 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.

10.1.7 Maximizing the Efficiency of Packets Sent Over the Network

Latency can be the most significant factor that influences the responsiveness of an application. 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 Server.

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. One way to avoid the latency concerns adhering to triggers is by grouping them together through Event Bundling. 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 Server.

Event Bundling gathers all of the events triggered while navigating between the two objects, and delivers them as a single packet to Oracle Application Server Forms Services for processing. 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 Application Server Forms Services as a single network message.

10.1.8 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 Application Server Forms Services be minimized in order to contain the latency effects of the network.

10.2 Tuning OracleAS Forms Services Applications

An application developer can take steps to ensure that maximum benefits are gained from Forms Server's 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 Server features.

10.2.1 Location of the Oracle Application Server Forms Services with Respect to the Data Server

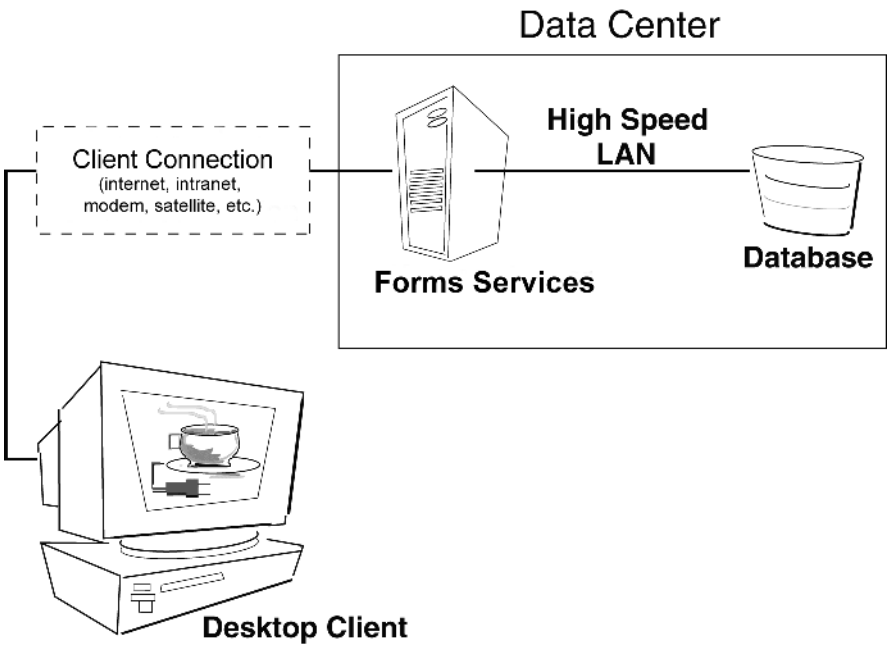
The Forms Java client is only responsible to display the GUI objects. All of the Oracle Forms logic runs in Oracle Application Server Forms Services, on the middle tier. This includes inserting or updating the data to the database, querying data from the database, executing stored procedures on the database, and so on. Therefore, it is important to have a high-speed connection 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, such as with modems or satellites.

The configuration in Figure 10–1, displays how Forms Services and the database server are co-located in a data center.

Figure 10–1 Co-Locating the OracleAS Forms Services and Database Server



10.2.2 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

10.2.2.1 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.

Oracle Application Server Forms Services provides the following pre-configured Jar files to support typical deployment scenarios.

10.2.2.1.1 Oracle JInitiator

The following are the Jar files provided for use with Oracle JInitiator:

- frmall.jar - includes all required classes
- frmall_jinit.jar - same as frmall.jar but is optimized for use with Oracle JInitiator (this is the default)
- frmmain.jar - contains fewer classes than frmall.jar. The other classes are downloaded as needed using a deferred mechanism. This gives a smaller download and a faster startup time.

To specify one or more Jar files, use the archive_jini setting in the named configuration section of the Forms Configuration file (formsweb.cfg). For example,

```
[MyApp]
archive_jini=frmall_jinit.jar, icons.jar
```

Your archive_jini setting must use only one of the three Jar files listed, above. It may also contain any additional custom Jar files that your application uses (for example, icons.jar, as shown in the previous example). Each application can use its own archive_jini setting.

The following Jar files contain the deferred classes that are missing from frmmain.jar. They will be downloaded automatically as they are needed, so there is no need to reference them in the archive_jini setting. They are already present in frmall.jar and frmall_jinit.jar, so they are only used if you use frmmain.jar.

- `frmoracle_laf.jar` – classes for the Oracle Look-And-Feel
- `frmgeneric_laf.jar` – classes for the generic (standard) Look-And-Feel
- `frmresources.jar` – resource classes for languages other than US English.

  The English resource classes are contained in `frmall.jar`, `frmall_jinit.jar`, and `frmmain.jar`. `frmresources.jar` will be loaded if a language other than US English is used. Note that this Jar file contains the resources for all languages other than English. Therefore you will have either the US English resource classes, or all of the language resource classes.

For more information about Oracle JInitiator, see Appendix B, "JInitiator".

10.2.2.1.2 All other cases (for example, Sun's Java Plug-in)

The following Jar file is provided for Java Virtual Machines (JVMs) other than Jinitiator or the IE native JVM:

- `frmall.jar` - includes all required classes

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
```

10.2.2.2 Using Caching

Both of the supported JVMs for Oracle Application Server Forms Services (Oracle JInitiator and Oracle JDK) support the caching of Jar files. 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.

10.2.3 Reducing the Required Network Bandwidth

The developer can design the application to maximize 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:

- **Control the order in which messages are sent.** The order in which messages are sent is governed by two criteria:
  - For the initial display, the display order in the Object Navigator
  - During execution, the order of program changes to item properties

  Where the result does not impact usability, you should strive to place similar objects that are on the same canvas after each other in the Object Navigator. For example, place buttons with buttons, text items with text items, and so on. (If you use the item property Next Navigation Item, the same order of navigation will be used for the items in the Form.) By ordering similar items together on the Object Navigator, the item properties sent to the client to display the first Form will include many similar items in consecutive order, which allows the *message diff-ing* algorithm to function efficiently.

In addition, when triggers or other logic are used to alter item properties, then you should group properties of similar items together before altering the item properties of another display type. For example:

```
set_item_property(text_item1_id, FONT_WEIGHT, FONT_BOLD);
set_item_property(text_item2_id, FONT_WEIGHT, FONT_BOLD);
set_item_property(text_item3_id, FONT_WEIGHT, FONT_BOLD);
set_item_property(button_item1_id, LABEL, 'Exit');
...
```

- **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)

    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.

10.2.4 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 filesystem and downloaded repeatedly to users' computers.

10.3 Web Cache and Forms Integration

Oracle Web Cache can be used as a load balancer with Oracle Forms applications.

The following setup instructions assume the following:

1. Oracle Application Server Web Cache instance running on Host A
2. Oracle HTTP Server instance and OC4J instance on Host B running Oracle Forms application D
3. Oracle HTTP Server instance and OC4J instance on Host C running Oracle Forms application D

Note that there could be more Oracle HTTP Server/OC4J instances, but only two instance pairs will be described here for purposes of simplification. The Oracle HTTP Server/OC4J instances are not clustered because Oracle Forms applications cannot take advantage of Oracle Application Server clustering.

Also note that a Web Cache 9.0.2.x cluster cannot be used. An Oracle Application Server Web Cache cluster can be used to load balance Oracle Forms starting with Oracle Application Server.

Since Forms applications are stateful, Web Cache must be configured for stateful load balancing using its session binding feature.

Configure Web Cache on Host A with the appropriate Site information for the Forms application, as well as Origin Server and Site-to-Server Mapping information for the Oracle HTTP Server instances running on Hosts B and C. When configuring Origin Server information for Hosts B and C, be sure to configure a ping URL that will detect whether Forms application D is running, for example, `/forms/frmservlet?ifcmd=status`.

To Configure Session Binding in Web Cache:

1. Add the following code to the orion-web.xml file located in $ORACLE_HOME\j2ee\OC4J_BI_Forms\application-deployments\formsapp\formsweb\orion-web.xml:

```
<session-tracking
 cookies="enabled">
</session-tracking>
```

2. Issue this command:

```
dcmctl updateconfig -ct oc4j
```

3. Restart OC4J_BI_Forms with:

   ```
   opmnctl restartproc gid="OC4J_BI_Forms"
   ```

4. Log on to the Web Cache Manager.
5. In the navigator pane, select **Origin Servers, Sites, and Load Balancing | Session Binding**.
6. In the **Session Binding** screen, select **Default Session Binding**, then select **Edit Selected**.
7. The **Edit Session Binding** dialog box appears.
8. From the **Please select a session:** pull-down list, select **Monitoring | Health Monitor**.
9. Configure an **Inactivity Timeout** that is appropriate for Oracle Forms application D.
10. Click **Submit**.
11. Apply changes and restart Oracle Application Server Web Cache.

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. Identify the Oracle HTTP Server/OC4J that handled the requests. For example, assume this is Host B and shut down the Oracle HTTP Server/OC4J on that host. Now only the Oracle HTTP Server/OC4J 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. Remember that Oracle Forms session state is not replicated among OC4J instances.
4. Next, use the browser to start a new Forms session. Web Cache will direct the requests to the remaining Oracle HTTP Server/OC4J running on Host C. Ensure that the application works as expected.
5. Restart the Oracle HTTP Server/OC4J on Host B. Using a browser, log on to the Web Cache Manager. In the navigator pane, select **Administration | Monitoring | Health Monitor**.
6. On the Health Monitor screen, make sure that Host B is marked **UP**.

For additional information about Web Cache, see *Oracle Application Server Web Cache Administration and Deployment Guide*.

11

Upgrading to OracleAS Forms Services

Oracle supports upgrading from Oracle6*i*AS Forms to Oracle Application Server 10*g*. Please read this chapter before you start the upgrade process. It contains the following sections:

- Section 11.1, "OracleAS Forms Services Upgrade Items"
- Section 11.2, "Components Related to OracleAS Forms Services"
- Section 11.3, "OracleAS Forms Services Upgrade Tasks"
- Section 11.4, "Validating the OracleAS Forms Services Upgrade"

If you're upgrading from Oracle9*i*AS Forms Services to OracleAS Forms Services, see the *Oracle Application Server Upgrade and Compatibility Guide* that is available on the Oracle Application Server 10*g* Documentation CD.

11.1 OracleAS Forms Services Upgrade Items

Table 11–1 describes the items that are upgraded. These items include files, executables, or settings that you must add, change, delete, or replace in the Oracle Application Server Forms Services installation.

Table 11–1 OracleAS Forms Services Upgrade Items

Upgrade Item	Location in 6i Oracle home	Location in 10*g* (10.1.2) Oracle home	Description and Notes
Oracle HTTP Server configuration file: `6iserver.conf` (upgrades to `forms.conf`)	`6iserver/conf/`	`/forms/server`	Contains virtual path mappings.
Servlet environment file: `default.env`	`6iserver/forms60/server`	`/forms/server`	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`	`/j2ee/OC4J_BI_Forms/ /applications/ formsapp/formsweb/ WEB-INF/web.xml`	Contains Forms servlet aliases.

Table 11–1 (Cont.) OracleAS Forms Services Upgrade Items

Upgrade Item	Location in 6i Oracle home	Location in 10*g* (10.1.2) Oracle home	Description and Notes
Application configuration file: `formsweb.cfg`	`6iserver/forms60/server`	`/forms/server`	Contains Forms Services application configuration information.
Forms servlet template html files: (`*.htm`, `*.html`)	`6iserver/forms60/server`	`/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 OracleAS Forms Services.

11.2 Components Related to OracleAS Forms Services

This section describes the relation between OracleAS Forms Services and other components. OracleAS Forms Services integration with Oracle Application Server is dependent on a set of OracleAS Forms Services configuration files. These dependencies are listed below in Table 11–2.

Table 11–2 Oracle Application Server Component and OracleAS Forms Services Configuration File Dependencies

Component	Configuration File
Oracle HTTP Server	`forms.conf`
Oracle Application Server Containers for J2EE	`web.xml`, `formsapp.ear`
Oracle Application Server Single Sign-On/Oracle Internet Directory	`formsweb.cfg`
Oracle Enterprise Manager 10*g*	`formsweb.cfg`
Oracle Reports	Forms internal PL/SQL built-in

11.3 OracleAS Forms Services Upgrade Tasks

This section explains how to perform the Oracle Application Server 10*g* Forms Services upgrade. It is divided into the following sub-sections:

11.3.1 Upgrade Recommendations and Troubleshooting Tips

Consider the following recommendations and considerations while upgrading Forms applications to Oracle Application Server 10*g*:

- Keep the Oracle6*i* Forms Services installation available until successful deployment and testing of applications to Oracle Application Server 10*g*.
- Upgrade source files first, and back up and secure all application 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 Application Server and configure the `forms/server/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 Application Server is installed, if it is not the same computer.
- After starting Oracle Application Server, access the Forms Services Listener Servlet test page with this URL:

  `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`*`>`

11.3.2 Upgrading OracleAS 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 OracleAS Forms Services installation.

1. Copy the Forms application files to a new directory.
2. Use the Forms Migration Assistant to upgrade the Forms Application Modules (`.fmb` files).
3. Use the Forms Migration Assistant to upgrade the Forms menu modules (`.mmx` files).
4. Use the Forms Migration Assistant to upgrade the library modules (`.pll` files) and Menu modules (`.mmb` files).
5. Use the Forms Migration Assistant to upgrade to upgrade the library modules (`.plx` files)
6. Use the Forms Compiler (`frmcmp.sh` on Unix or `frmcmp.exe` on Windows) to regenerate the Forms Application executable files (`fmx` files).

For more information, see *Migrating Forms Applications from Forms 6i* at:

`http://www.oracle.com/technology/documentation/`

11.3.3 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 OracleAS 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 `forms/server/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 OracleAS Forms Services configuration, `formsapp` and `formsweb`, in the Oracle Application Server Containers for J2EE Business Intelligence and Forms instance (`<destination_MT_OH>/j2ee/oc4j/OC4J_BI_Forms`).

Follow these steps to upgrade an Oracle 6*i* Forms Services Release 6*i* CGI environment to an OracleAS Forms Services servlet environment:

1. Copy all of the application-specific configurations from `<source_MT_OH>/Forms60/Server/formsweb.cfg` and append them to `<destination_MT_OH>/forms/server/formsweb.cfg`.

   Note: Do not copy and replace the entire `formsweb.cfg` file in `<source_MT_OH>` to `<destination_MT_OH>`. The file in Release 6*i* is different from the OracleAS Forms Services file. Copy only the application configuration to `<destination_MT_OH>/forms/server/formsweb.cfg`.

2. Configure `Forms_Path` in the `forms/server/default.env` file to point to the upgraded OracleAS 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 OracleAS Forms Services HTML template files.

> **Note:** You must make these changes in `basejini.htm` and `basejpi.htm` because the servlet supports JInitiator and Java plug-ins.

11.3.4 Upgrading Static HTML Start Files to Generic Application HTML Start Files

Each application deployed to OracleAS Forms Services has a custom application definition, configured in the `formsweb.cfg` configuration file. It automatically inherits the general system settings, such as the JInitiator version used or 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 OracleAS 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 JInitiator or another VM, but not Internet Explorer native.

```
[booking_app]
archive_jini=frmall_jinit.jar, custom.jar
archive=frmall.jar, custom.jar
ServerURL=/forms/lservlet
Form = book.fmx
lookAndFeel=oracle
Otherparams=
```

```
Userid=
```

Follow these steps to upgrade applications:

1. Edit the `forms/server/default.env` file, adding the location of the OracleAS Forms Services application modules to the `Forms_Path`.
2. Edit the `forms/server/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>`

11.3.4.1 Using Static HTML Files with OracleAS Forms Services

If you need to, you can continue to use static HTML files in OracleAS Forms Services. However, with static HTML files, some features (such as Oracle Application Server 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 OracleAS 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. OracleAS Forms Services uses JInitiator version 1.3.x, so you must also change those settings. The required values are found in the `forms/server/formsweb.cfg` file.

Follow these steps to use static HTML files with OracleAS Forms Services:

1. Configure `Forms_Path` in the `forms/server/default.env` file to point to the upgraded OracleAS Forms Services application modules.
2. Create virtual directories in the *<destination_MT_OH>*`/forms/server/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`.
   b. Change the JInitiator version number.
4. Change the `codebase` parameter to `forms/java`.
5. Navigate to *<destination_MT_OH>*`/j2ee/OC4J_BI_Forms/applications/formsapp/formsweb/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 *<destination_MT_OH>*`/forms/server/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_MT_OH/forms/server/default.env</param-value>
```

```
  </init-param>
</servlet>
```

11.3.5 Upgrading the Forms 6i Listener to the Forms Listener Servlet

The Forms 6*i* Listener is a C program that starts a Forms runtime process on behalf of an incoming Forms Web request. The Forms Web runtime process is then directly accessed by the Forms client applet, using a direct socket or an HTTP socket connection. The Forms Listener is then no longer involved in the application Web client-server communication process, and is 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 server-side 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 listens on a specific port (by default, 9000), the Forms Servlet doesn't 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 Oracle Application Server installation. The installer creates a virtual path `/forms/` pointing to the OracleAS Forms Services configuration, `formsapp` and `formsweb`, in the Oracle Application Server Containers for J2EE Business Intelligence and Forms instance (`<destination_MT_OH>/j2ee/oc4j/OC4J_BI_Forms`).

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 `forms/server/formsweb.cfg` file. All application configurations automatically inherit the `serverURL` parameter value `/forms/lservlet` from the global system parameter settings.

Note: Do not copy and replace the entire `formsweb.cfg` file in `<source_MT_OH>` to `<destination_MT_OH>`. The file in Release 6*i* is different from the OracleAS Forms Services file in 10*g* Release 2 (10.1.2). Copy only the application configuration to `<destination_MT_OH>/forms/server/formsweb.cfg`.

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 OracleAS Forms Services modules as described in Section 11.3.2, "Upgrading OracleAS Forms Services Application Modules". on page 11-3.

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_MT_OH*>`/Forms60/Server/formsweb.cfg` and append them to <*destination_MT_OH*>`/forms/server/formsweb.cfg`.

   Note: Do not copy and replace the entire `formsweb.cfg` file in <*source_MT_OH*> to <*destination_MT_OH*>. The file in Oracle 6*i* Forms Services is different from the OracleAS Forms Services file. Copy only the application configuration to <*destination_MT_OH*>`/forms/server/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_MT_OH*>`/forms/server/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 OracleAS 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
   baseHTMLJinitiator=<your base jinit>.htm
   ```

6. Start the application with this URL:

   `http://<`*hostname*`>:<`*port*`>/forms/frmservlet?config=<`*application*`>`

11.3.6 Upgrading the Forms Listener Servlet Architecture to OracleAS 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 OracleAS Forms Services, the Forms Listener servlet is the same except for the servlet names, which are `frmservlet` and `lservlet`, and the servlet container, which is now Oracle Application Server Containers for J2EE (OC4J). As in Oracle9*i*AS Release 1 (1.0.2.2.x), the configuration is performed during installation. The Listener Servlet configuration in OC4J is stored in `ORACLE_HOME/j2ee/OC4J_BI_Forms/applications/formsapp/formsweb/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 Application Server installation. The installer creates a virtual path `/forms/` pointing to the OracleAS Forms Services configuration, `formsapp` and `formsweb`, in the Oracle Application Server Containers for J2EE Business Intelligence and Forms instance (`<destination_MT_OH>/j2ee/oc4j/OC4J_BI_Forms`).

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 OracleAS Forms Services:

1. Copy the Forms application files to a new directory and upgrade them to OracleAS 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_MT_OH>/Forms60/Server/formsweb.cfg` and append them to `<destination_MT_OH>/forms/server/formsweb.cfg`.

   Note: Do not copy and replace the entire `formsweb.cfg` file in `<source_MT_OH>` to `<destination_MT_OH>`. The file in Release 6*i* is different from the OracleAS Forms Services file in 10*g* Release 2 (10.1.2). Copy only the application configuration to `<destination_MT_OH>/forms/server/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_MT_OH>/forms/server/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 OracleAS 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
baseHTMLJinitiator=<your base jinit>.htm
```

6. Start the application with this URL:

   ```
   http://<hostname>:<port>/forms/frmservlet?
   config=<application>
   ```

11.3.7 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 OracleAS Forms Services, load balancing is managed by `mod_oc4j`, an Oracle HTTP Server module. It binds Web requests to the servlet container processing the Forms Servlet and the Forms Listener servlet.

11.3.8 Usage Notes

This section contains hints and tips that may be useful in the upgrade.

11.3.8.1 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 OracleAS 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 (e.g.,.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 lines to the custom application section:

```
archive_jini=frmall_jinit.jar,<application>_images.jar
archive_frmall.jar,<application>_images.jar
```

See Section 4.9, "Deploying Icons and Images Used by Forms Services" for more information on deploying custom icon files with OracleAS Forms Services.

11.3.8.2 Upgrading Integrated Calls to Oracle Reports to use Oracle Reports

In Oracle Application Server, integrated calls to Oracle Reports in Forms are no longer handled by a client-side background engine. OracleAS 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 Application Server Reports Services Publishing Reports to the Web* and http://www.oracle.com/technology/products/forms/pdf/10g/frmreppparamform.pdf

11.3.8.3 Creating Forms Listener Servlet Alias Names in OC4J

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 OracleAS 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 <*destination_MT_OH*>/j2ee/OC4J_BI_Forms/applications/formsapp/formsweb/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>
```

11.3.8.4 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_MT_OH>/Apache/Apache/JServ/conf/zone.properties` file.

You can display the test page for OracleAS 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 `<destination_MT_OH>/j2ee/OC4J_BI_Forms/applications/formsapp/formsweb/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>
```

11.4 Validating the OracleAS Forms Services Upgrade

After you complete the upgrade tasks, ensure that the upgraded version of the OracleAS 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 Oracle9*i*AS 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 Application Server 10*g*, the forms application URL is typically:

```
http://<hostname>:<port>/forms/<forms servlet alias>?<forms
application name>
```

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. Fortunately, after the Oracle Universal Installer says that Oracle Application Server or Developer Suite was successfully installed, you can be verify whether Oracle Forms services is correctly configured or not. You can use these tools as solutions:

A.1.1 Use The Web Form Tester

The Form Tester is available with your Oracle Application Server or Developer Suite installation. To verify whether the OracleAS 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 Oracle HTTP Server by selecting **Start | Program Files Oracle Application Server-AS Home | Oracle HTTP Server | Start HTTP Server**, if it is not already started.

For Oracle Developer Suite, start the OC4J instance (if not already started) by selecting **Start | Programs | Oracle Developer Suite - DevSuiteHome | Forms Developer | Start OC4J Instance**.

2. Open an instance of the browser by typing `ORACLE_HOME/tools/web/html/runform.htm` for the URL and hit the ENTER key. Replace ORACLE_HOME with your actual Oracle home for OracleAS or Developer Suite.
3. You may also run the Web Form Tester by selecting **Start | Program Files | OracleAS–Home for AS | Forms Developer | Run a Form on the Web** from the Windows Start menu for OracleAS.

   For Oracle Developer Suite, select **Start | Program Files | Oracle Developer Suite–Home for DS | Forms Developer | Run a Form on the Web** from the Windows Start menu.
4. 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.
5. If the installation of OracleAS or Developer Suite is correct, you'll see a success message in the Web browser. Also, it can be tested from a client computer whether the basic Forms setup in OracleAS or Developer Suite 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://myserver.com: NNNN/forms90/f90servlet?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"
- Section A.2.2, "The JInitiator Java Console"
- Section A.2.3, "FRM-92XXX Error Messages"

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.2.2 The JInitiator Java Console

If you are using JInitiator and a Java error is encountered then the error will be written to the browser status line. However, this does not show the full Java error stack. You need to look for it in the JInitiator Java console.

If you have turned the Java Console on, by checking its option in JInitiator Control Panel Applet on a Windows computer, the JInitiator Console window will pop up when your Form runs in a browser. The JInitiator Control Panel Applet can be found on the **Start | Settings | Control Panel** option in Windows. If JInitiator's Java console does not appear, you can always invoke it manually from the taskbar tray icon by double clicking it.

A.2.3 FRM-92XXX Error Messages

While running your Forms application, you may encounter various FRM errors. These errors can be raised by several different conditions. When you receive these errors, you will need to obtain more information to resolve it. In this section we will see some of the common errors that one may encounter and how to resolve them.

Broadly speaking, the FRM errors occur due to the following.

- Configuration Problems:

  Some FRM errors are raised by configuration problems. For example, the Forms Service is not started, or is listening on a different port to that specified in the HTML file. Typically these errors will reproduce consistently.

- Forms server process has crashed:

  The majority of FRM errors that occur after a successful connection has been established and the form started, are due to the server crashing. Once the server process has died, then the client cannot continue running - the applet has no life of its own, and it cannot continue to run without being able to communicate with the server process.

  These errors are often difficult to diagnose: the problem may not reproduce consistently, or the user may be unaware of the sequence of events that led to the crash.

- Network Problems:

  The communication between the applet and the Forms Server process has experienced network problems, and the communication has been broken.

The following table lists the FRM -92xxx errors caused by these reasons. It also briefly explains what they mean.

Error	Description
FRM-92000	This is an internal error that occurs when the Java language throws an IllegalAccessException whilst we are trying to load some class files. It usually indicates that the system is misconfigured in some way. The detail message often helps to work out why it occurred.
FRM-92010	This is a Client mis-configuration error that occurs when the Applet parameter "serverArgs" is either not present or has a null value.
FRM-92020	Indicates that either the URL, or the Browser Target, requested was rejected in some way by the browser.

Error	Description
FRM-92030	A Client mis-configuration error, due to a missing Java class file and/or registry mis-configuration. This error occurs when the Server requests a Java class, by numeric "handlerClassId" that the client can't handle since it's not in the registry.
FRM-92040	A Server mis-configuration error, due to a missing Java class file. This error occurs when the Client requests a Java class that couldn't be located on the server.
FRM-92050	The Client was unable to establish a connection to the server computer (host) on the designated socket (port).
FRM-92060	The Client was unable to establish a connection to the Server because the format of the host/port combination was invalid.
FRM-92070	The Client was unable to create a new Object for some reason. The Full details may give some indication as to why the error occurred. (This will not stop the working of the form, it is logged only in the log file)
FRM-92080	Executing an Operating System command, in an attempt to start an external Browser module, caused some problem.
FRM-92090	An Unexpected error occurred.
FRM-92095	The version of JInitiator being used is too low to support the requested functionality (e.g. to run against the Listener Servlet). User should install the specified version (or greater).
FRM-92100	An Unexpected Network error or server failure occurred.
FRM-92101	An unexpected server failure occurred due to some misconfiguration on server side.
FRM-92102	An Unexpected Network error occurred, after trying to reconnect for a specific number of times by Forms.
FRM-92120	A Server configuration error that indicates that an important file (the Registry) could not be located by the client.
FRM-92145	The text used to describe Single Sign-On Authentication failed.
FRM-92150	The version of the client is newer than the version of the server.
FRM-92160	The version of the client is older than the version of the server.

Error	Description
FRM-93000	Generic internal Exception message. Used when some unexpected code error occurs.

While most of the above FRM errors are self-explanatory, there are a few which are caused for different reasons and are difficult to diagnose. The following topic explains such FRM errors, the possible causes for them, and their solutions.

A.2.3.1 FRM-92010

Cause:

This error can occur when JInitiator uses the browser's proxy setting.

Solution:

Go to Control Panel | JInitiator 1.3.x.x | Proxies and deselect Use Browser Settings and enter the details for the proxy settings.

A.2.3.2 FRM-92050

- Heavy load on the server.

  Cause:

  If there are many simultaneous requests that the server cannot handle. This mainly depends on the server computer performance and configuration.

  Solution:

  - The Forms Runtime Prestart feature of Oracle Application Server Forms Services 10*g* comes in handy in this situation. This feature pre-spawns a configurable number of runtime engines to handle incoming client requests, and avoiding application or server hangs because of a rush.
  - Upgrade the hardware of the server computer to handle the high number of simultaneous requests.

- Missing serverURL Parameter

  Cause:

  The serverURL parameter is either missing or incorrect, in the configuration file (formsweb.cfg)

  Solution:

  Edit the forms configuration file to enter a valid serverURL parameter value.

- Wrong `FORMS_TIMEOUT`

  Cause:

  The value of `FORMS_TIMEOUT` parameter is entered wrongly.

  Solution:

  Verify the environment file (default.env) and the registry for the `FORMS_TIMEOUT` parameter value. The value should be a proper integer. The value should not be in quotes, for example:

  `FORMS_TIMEOUT="10"` This is an incorrect entry.

FORMS_TIMEOUT=10 This is the correct entry.

- Incorrect Use of RUN_PRODUCT

  Cause:

  RUN_PRODUCT should only be used, in Oracle Forms, for integration with Oracle Graphics *6i*.

  Solution:

  RUN_PRODUCT Built-in calls that are used to integrate Oracle Forms with Oracle Reports should be replaced using the newer RUN_REPORT_OBJECT Built-in.

- Missing ServerArgs parameter

  Cause:

  The ServerArgs parameter is missing from the HTML, which loads the applet.

  Solution:

  - Make sure that the HTML file, used to load the forms applet, has the ServerArgs parameter in it.
  - Make sure that the value of the ServerArgs is not null. Remember, the Form name is required in ServerArgs. These parameters can be defined in the Oracle Forms configuration file (formsweb.cfg) or can be directly passed in the URL that is used to run the Form.

- Missing jvm.dll

  Cause:

  The Forms Web executable frmweb.exe is not able to find the jvm.dll

  Solution:

  Ensure that jvm.dll is located in a directory specified in the PATH environment variable. Set the PATH environment variable in formsweb.cfg, which is typically ORACLE_HOME/forms/server/default.env, to point to the location of the jvm.dll.

A.2.3.3 FRM-92100

Cause:

This error occurs if the Web server is shutdown when the user is accessing the application

Solution:

Check if the Web server is up and running. Try the URL http://servercomputer:portno. If the OC4J home page does not come up, then it indicates that the Web server is down. Contact your Forms or server administrator to start the Web server.

A.2.3.4 FRM-92101

- Wrong working directory

  Cause:

  This error can occur if the working directory specified does not exist.

  Solution:

- This can be confirmed by looking for a log message like "Unable to switch to Working Directory:<workingDirectory>" in the application.log file. The application.log file can be found in the `application-deployments/formsapp` directory of the OC4J instance on which Oracle Forms is deployed.

  Edit the forms configuration file with the correct working directory.

- `FORMS_TIMEOUT` and `heartbeat`

  Cause:

  This error can occur if the forms applet parameter 'heartbeat' is set to a value more than `FORMS_TIMEOUT`.

  Solution:

  Generally, heartbeat is set to a value higher than that of `FORMS_TIMEOUT`, only when the application is desired to time-out after a certain time of inactivity. It is then, you would get a FRM -92120.

  If that is not desired for a particular application, then, make sure that the value of heartbeat is less than that of `FORMS_TIMEOUT`.

A.2.3.5 FRM-92102

Cause:

This error can occur because of the network problems between the Web server and the client. The client is not able to communicate with the server on the specified port.

Solution:

Add the parameter `networkRetries` to the forms configuration file. Set the value to a suitable number according to the network characteristics and needs, for example, `networkRetries=30` This parameter specifies the number of times the Forms client should try reconnecting to the middle tier before finally timing out.

- Ports Blocked

  Cause:

  If the error occurs even after setting up an appropriate value for networkRetries, it could be due to the ports on the web server restricted at TCP/IP level.

  Solution:

  A socket connection requires a port at each end. If the port is closed it causes the communication stoppage. Firewall and proxies are used to protect the ports. Removing the blocks on the ports on the Web server solves the error.

A.2.3.6 FRM-92120

Cause:

This is a server configuration error, which occurs when the client is unable to find the file Registry.dat on the middle tier.

Solution:

When this error occurs, check if the file Registry.dat is present on the middle tier in the directory `ORACLE_HOME/forms/java/oracle/forms/registry`. If it is not present, then it needs to be placed.

In a running Forms application, if you suddenly come across this error, there is a possibility that the HTTP server has gone down. You may verify this by typing the

URL `http://myserver.com:NNNN` in your browser. Here you need to replace myserver.com with your host name and NNNN with your HTTP server's port number. If your browser says that it could not connect to the server, then your HTTP server is down and you need to contact your system administrator to bring it up.

When the HTTP server is up and running, on giving the URL http://myserver.com:NNNN your browser will show the "OracleAS welcome".

A.2.3.7 FRM-92150/FRM-92160

Cause:

Wrong path and/or codebase setting.

Solution:

Set the proper `ORACLE_HOME/bin` in the beginning of the system path. The `CODEBASE` entry in your HTML file or forms configuration file may point to older versions of the Jar file. Either modify the codebase entry in your configuration file or replace the jar file in the codebase path with the appropriate jar file.

Clearing the Oracle Jar cache in the user profile directory of the client computer makes sure that the fresh Forms Jar files are downloaded.

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_HOME/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.

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 Solaris 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. The stack above was generated with FORMS_DELIBERATECRASH set to `2`. This environment variable can be set in the environment (for example, default.env) file.

By setting the environment variable FORMS_DELIBERATECRASH to `2`, and checking the stack trace produced against the one in this document you can determine whether the symbol files are correctly installed. Once you have confirmed that everything is working as expected, if you subsequently encounter a problem where the server has crashed, you can be sure that the stack trace will be useful in resolving the problem.

A.3.2.2 Understanding Solaris Stack Traces

In a Solaris 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.

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 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:

- Section A.4.1, "About Diagnosing Client Crashes"
- Section A.4.2, "Diagnosing Hanging Applications"

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. If you're using Appletviewer that was started from the directory `c:\jdk\1_3_1\bin`, the client stack trace file will be `c:\jdk\1_3_1\bin\appletviewer.rpt`.

If you're using JInitiator, then the executable is considered to be the Web browser. If the browser is Netscape, the client stack trace file will be `netscape.rpt`, whereas for Internet Explorer it will be `iexplore.rpt`.

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 Server process 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.

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 server version. Communication between the applet and the Forms server process is based on message ID. If these message ID's are out of synch, 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. The timeout facility acts like a heartbeat and expects the Oracle Forms client to 'ping' the server on a set interval, only cleaning up the Oracle Forms Server process when there has been no activity from the Forms client for the specified time. Although this is primarily intended to prevent orphaned server processes, it can also prevent the unwanted premature cleanup of server processes.

A.4.2.2 Creating Java Thread Dumps

A stack dump can be obtained from an Appletviewer by pressing CTRL+BREAK in the command prompt (or DOS session) that you started the Applet-viewer from.

For JInitiator, the Java console shows how to output dump information. Follow the onscreen instructions to view it.

The information contained in the thread dump can help Oracle development identify the problem in the code. The thread dump should be included in any bug filed to report the problem.

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 8.1, "About Forms Trace" and Chapter 8.7, "Servlet 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"

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 Enterprise Manager or at the command line in the middle tier. For example, the following command will set the initial 'Java Heap' (the memory used by the JVM) to 20MB, and the maximum memory to be used by the JVM to 32MB:

```
appletviewer -J-ms20m -J-mx32Mb dve.html
```

The default initial and maximum sizes for Appletviewer are 16MB and 20MB respectively.

When using JInitiator, you will need to set the runtime options in the JInitiator control panel.

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.

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 (GC). 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. 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.

In addition to reducing the amount of data to be transferred, Jar files also allow JInitiator and Oracle's JDK to use a process referred to as caching. Starting with OJDK 1.1.7.15, several performance improvements were made to the caching process.

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.6.5 Performance Improvements in OJDK

While caching means that the Jar file will not be downloaded from the server every time the application is invoked, there were still some issues affecting performance. Since the Jar files contain compressed data, the time to decompress this data from the cached Jar file outweighs the time saved in downloading less data from the network.

Jar files can be digitally signed and undergo authentication to ensure they have not been modified in transit over the network. This procedure involves expensive mathematical calculations. The new caching mechanism introduced in OJDK addresses these issues in the following way:

When the Jar file is downloaded for the first time, two files are created:

- A data file, which contains all of the unzipped data from the Jar file. Data files have the extension .dxx, where xx is a number between 00 and 99. For example 10f756b8.d00.
- An index file which contains information about the Jar file, including the URL it was loaded from, the date it was last modified on the server, and a table of contents. The table of contents lists all of the entries in the Jar file, their offsets in the data file, and the authentication status of each entry. Index files have the extension .ixx, where xx is a number between 00 and 99. For example 10f756b8.i00.

The information in these files is stored in a binary format. There is no easy way to read them by eye, and there is little value in doing so.

The first eight characters of all cache files represent the URL where the Jar file was downloaded from. This allows the caching mechanism to quickly find a URL in the cache by mapping the URL to its corresponding eight-character representation and looking for files with that name.

When a class file is required by the application, OJDK uses the information in the table of contents to locate the class file in the data cache. If the data was digitally signed, then the list of authenticated signers is read from the data file.

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

Don't 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.

Don't 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?

In case the information in the previous sections was not sufficient, you can find more solutions on Oracle *MetaLink*, `http://metalink.oracle.com/`. If you do not find a solution for your problem, log a service request.

See Also:

- *Oracle Application Server Release Notes*, available on the Oracle Technology Network: `http://otn.oracle.com/documentation/ias.html`

B

JInitiator

This section describes the benefits of using Oracle JInitiator as a Web browser plug-in. Oracle JInitiator enables users to run Oracle Forms applications using Netscape Navigator or Internet Explorer. It provides the ability to specify the use of a specific Java Virtual Machine (JVM) on the client, rather than using the browser's default JVM.

Oracle JInitiator runs as a plug-in for Netscape Navigator and as an ActiveX component for Internet Explorer. Oracle JInitiator does not replace or modify the default JVM provided by the browser. Rather, it provides an alternative JVM in the form of a plug-in.

Oracle provides two Jar files (`frmall.jar` and `frmall_jinit.jar`). `frmall.jar` is a standard Jar file, and `frmall_jinit.jar` is a Jar file with extra compression that can only be used with Oracle JInitiator.

B.1 Why Use Oracle JInitiator?

Oracle JInitiator delivers a certified, supportable, Java Runtime Environment (JRE) to client desktops, which can be launched transparently through a Web browser.

Oracle JInitiator is Oracle's version of JavaSoft's Java Plug-in. The JavaSoft Plug-in is a delivery mechanism for a JavaSoft JRE, which can be launched from within a browser. Likewise, Oracle JInitiator is providing a delivery mechanism for an Oracle certified JRE, which enables Oracle Forms applications to be run from within a browser in a stable and supported manner.

In addition to providing a certified platform for the execution of Oracle Forms applications, Oracle JInitiator provides a number of additional features over and above the standard JavaSoft Java Plug-in. These include Jar file caching, incremental Jar file loading, and applet caching (see Chapter 8, Minimizing the Application Startup Time).

B.2 Benefits of Oracle JInitiator

Oracle JInitiator provides these benefits:

- It allows the latest Oracle-certified JVM to run in older browser releases.
- It ensures a consistent JVM between different browsers.
- It is a reliable deployment platform. JInitiator has been thoroughly tested and certified for use with Forms Services.
- It is a high-performance deployment environment. Application class files are automatically cached by JInitiator, which provides fast application start-up.

- It is a self-installing, self-maintaining deployment environment. JInitiator automatically installs and updates itself like a plug-in or an Active-X component. Locally cached application class files are automatically updated from the application server.

B.3 Using Oracle JInitiator

The first time the client browser encounters an HTML file that specifies the use of Oracle JInitiator, it is automatically downloaded to a client computer from the application server. It enables users to run Oracle Application Server Forms Services and Graphics applications directly within Netscape Navigator or Internet Explorer on the Windows 98, NT, 2000, and XP platforms.

The installation and updating of Oracle JInitiator is performed using the standard plug-in mechanism provided by the browser. Oracle JInitiator installation performs the required steps to run Oracle Forms applications as trusted applets in the Oracle JInitiator environment.

B.4 Supported Configurations

Oracle JInitiator supports the following configurations:

B.4.1 Windows 98, NT, 2000, XP:

- Navigator 4.7.x
- Navigator 7.x
- Internet Explorer 5.x
- Internet Explorer 6.0

B.5 System Requirements

The minimum system requirements for Oracle JInitiator are:

- Windows 98, NT, 2000, XP
- Pentium 90 MHz or better processor
- 25MB free hard disk space (recommended 30MB)
- 16MB system RAM (recommended 32MB)

Note: These minimum system requirements are for JInitiator only; they are insufficient to run Oracle Forms.

B.6 Using Oracle JInitiator with Netscape Navigator

Oracle JInitiator leverages the Netscape Navigator plug-in architecture in order to run inside the browser in the same way other plug-ins, such as QuickTime movies or Shockwave animations operate. Using the Netscape HTML `<EMBED>` tag, Web application developers can specify that plug-ins run as part of a Web page. This is what makes it possible for Oracle JInitiator to run inside the Web browser with minimal user intervention.

When Navigator first encounters an HTML page that specifies the use of Oracle JInitiator, users will see a "Plug-in Not Loaded" dialog on the HTML page, which

directs the user to the Oracle JInitiator download page. Users can then download the version of Oracle JInitiator for their operating system and install it.

Once Oracle JInitiator is installed, users must shut down Navigator, restart it, and then revisit the original HTML page. Oracle JInitiator will then run and use the parameters in the <EMBED> tag to render the applet. The next time Navigator encounters a Web page that specifies Oracle JInitiator, Navigator will seamlessly load and run the plug-in from the local disk, without user intervention.

B.7 Using Oracle JInitiator with Microsoft Internet Explorer

Oracle JInitiator leverages the Microsoft Internet Explorer extension mechanism for downloading and caching ActiveX controls and COM components. Using the HTML <OBJECT> tag, Web application developers can specify that ActiveX controls or COM components should run as part of a Web page. Such components include Oracle JInitiator.

When Internet Explorer first encounters an HTML file that has been modified to specify the use of Oracle JInitiator, Internet Explorer will ask the user if it is okay to download an ActiveX control signed with a VeriSign digital signature by Oracle. If the user clicks "Yes," Internet Explorer will begin downloading Oracle JInitiator. Oracle JInitiator will then run and use its parameters in the <OBJECT> tag to render the applet. The next time Internet Explorer encounters a Web page modified to support Oracle JInitiator, it will seamlessly load and run Oracle JInitiator from the local disk, without user intervention.

B.8 Setting up the Oracle JInitator Plug-in

You can setup the Oracle JInitiator Plug-in after you install it on your server for server-based testing purposes only. Then continue by:

- Adding Oracle JInitiator Markup to Your Base HTML File
- Customizing the Oracle JInitiator Download File
- Making Oracle JInitiator Available for Download

B.8.1 Adding Oracle JInitiator Markup to Your Base HTML File

To add Oracle JInitiator markup to your base HTML file:

1. Open your base HTML file within a text editor.
2. Add the OBJECT and EMBED tags.

   For examples of added markup, refer to Appendix C.3, "base.htm, basejini.htm, and basejpi.htm Files".

B.8.2 Customizing the Oracle JInitiator Download File

The Oracle JInitiator download file (JINIT_DOWNLOAD.HTM) is the template HTML file that allows your users to download the Oracle JInitiator file.

To customize the Oracle JInitiator download file:

1. Open the JINIT_DOWNLOAD.HTM file within an HTML or text editor.
2. Modify the text as desired.
3. Save your changes.

B.8.3 Making Oracle JInitiator Available for Download

To make Oracle JInitiator available for download:

1. Copy jinit13*x*.EXE to your Web server.

   You must copy jinit13*x*.EXE to the location that was specified within the base HTML file.

2. Copy JINIT_DOWNLOAD.HTM to your Web server.

   You must copy JINIT_DOWNLOAD.HTM to the location that was specified within the base HTML file.

B.9 Modifying the Oracle JInitiator Plug-in

Continue modifying the Oracle JInitiator Plug-in by:

- Modifying the Cache Size for Oracle JInitiator
- Modifying the Heap Size for Oracle JInitiator
- Checking and Modifying the Proxy Server Setting for Oracle JInitiator
- Viewing Oracle JInitiator Output

B.9.1 Modifying the Cache Size for Oracle JInitiator

The default cache size for Oracle JInitiator is 20000000. This is set for you when you install Oracle JInitiator.

To modify the cache size for Oracle JInitiator:

1. From the Windows Start menu, choose **Start | Settings | Control Panel | Oracle JInitiator**.
2. Click the **Basic** tab.
3. In the **Java Run Time Parameters** field, specify the Dcache size. For example, specifying Dcache.size=20000000 sets the cache size to 20MB.

B.9.2 Modifying the Heap Size for Oracle JInitiator

The default maximum heap size for Oracle JInitiator is 64MB. This has been set for you when you install Oracle JInitiator.

To modify the heap size for Oracle JInitiator:

1. From the Windows Start menu, choose **Start | Settings | Control Panel | Oracle JInitiator**.
2. Click the **Basic** tab.
3. In the **Java Run Time Parameters** field, specify the mx size. For example, specifying mx64m means setting maximum heap size to 64MB.

B.9.3 Checking and Modifying the Proxy Server Setting for Oracle JInitiator

To check and modify the proxy server setting for Oracle JInitiator:

1. From the Windows Start menu, choose **Start | Settings | Control Panel | Oracle JInitiator**.
2. Click the **Proxies** tab.
3. Select the **Use Browser Settings** checkbox to allow **Oracle JInitiator** to use the settings in your browser's configuration dialog box. If you want to use another proxy server setting, be sure the box is not selected. Then, enter the host name for the proxy server in the **Proxy Address** field.

B.9.4 Viewing Oracle JInitiator Output

To view Oracle JInitiator output:

1. From the Windows Start menu, choose **Start | Settings | Control Panel | Oracle JInitiator**.
2. Click the **Basic** tab.
3. Select **Show Java Console** to enable debug output.

B.10 Modifying the baseHTML file

When you run an Oracle Forms application with the help of JInitiator, JInitiator reads parameter values from the formsweb.cfg file and passes these values into the baseHTML file. If you want to create a static baseHTML file so that the same values are read all the time, you need to manually place them in the baseHTML file.

For an example of the Oracle JInitiator markup for both Microsoft Internet Explorer and Netscape Navigator, see Appendix C.3, "base.htm, basejini.htm, and basejpi.htm Files". Adding these tags to your baseHTML file will enable your applications to run within both Netscape and Microsoft browsers.

C

Sample Configuration Files

During the installation, the following configuration files were installed onto your system:

C.1 Default formsweb.cfg File

The default formsweb.cfg file contains the following:

```
# 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.mydomain.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 JInitiator client
baseHTMLjinitiator=basejini.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: working directory for Forms runtime processes
# WorkingDirectory defaults to <oracle_home>/forms if unset.
workingDirectory=
# 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=
# Other Forms runtime arguments: grouped together as one parameter.
# These settings support running and debugging a form from the Builder:
otherparams=buffer_records=%buffer% debug_messages=%debug_messages% array=%array%
obr=%obr% query_only=%query_only% quiet=%quiet% render=%render% record=%record%
 tracegroup=%tracegroup% log=%log% term=%term%
# Sub argument for otherparams
buffer=no
# Sub argument for otherparams
debug_messages=no
# Sub argument for otherparams
array=no
# Sub argument for otherparams
obr=no
# Sub argument for otherparams
query_only=no
# Sub argument for otherparams
quiet=yes
# Sub argument for otherparams
render=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 Application Server 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=DocumentBase
# Forms applet parameter
width=750
# Forms applet parameter
height=600
# Forms applet parameter
separateFrame=false
# Forms applet parameter
splashScreen=
# Forms applet parameter
background=
# Forms applet parameter
lookAndFeel=Oracle
# Forms applet parameter
colorScheme=teal
# Forms applet parameter
logo=
# Forms applet parameter
restrictedURLparams=HTMLbodyAttrs,HTMLbeforeForm,pageTitle,HTMLafterForm,log,
allow_debug,allowNewConnections
# Forms applet parameter
formsMessageListener=
# Forms applet parameter
recordFileName=
# Forms applet parameter
serverApp=default
# Forms applet archive setting for JInitiator
archive_jini=frmall_jinit.jar
# 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 Netscape users to allow them to download Oracle JInitiator.
# Oracle JInitiator is used with Windows clients.
# If you create your own page, you should set this parameter to point to it.
jinit_download_page=/forms/jinitiator/us/jinit_download.htm
# Parameter related to the version of JInitiator
jinit_classid=clsid:CAFECAFE-0013-0001-0022-ABCDEFABCDEF
# Parameter related to the version of JInitiator
jinit_exename=jinit.exe#Version=1,3,1,22
# Parameter related to the version of JInitiator
jinit_mimetype=application/x-jinit-applet;version=1.3.1.22

# 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/1.4.2_06/index.html
# Parameter related to the version of the Java Plugin
```

```
jpi_classid=clsid:CAFEEFAC-0014-0002-0006-ABCDEFFEDCBA
# Parameter related to the version of the Java Plugin
jpi_codebase=http://java.sun.com/products/plugin/autodl/
jinstall-1_4_2-windows-i586.cab#Version=1,4,2,06
# Parameter related to the version of the Java Plugin
jpi_mimetype=application/x-java-applet;jpi-version=1.4.2_06
# EM config parameter
# Set this to "1" to enable Enterprise Manager to track Forms processes
em_mode=0

# Single Sign-On OID configuration parameter
oid_formsid=%OID_FORMSID%
# Single Sign-On OID configuration parameter
oracle_home=%ORACLE_HOME%
# Single Sign-On OID configuration parameter
formsid_group_dn=%GROUP_DN%
# 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
# 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=

# EndUserMonitoringURL
# indicates where to record EUM/Chronos data
EndUserMonitoringURL=

# 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.mydomain.com/forms/frmservlet?config=sepwin".
[sepwin]
```

```
separateFrame=True
lookandfeel=Generic

# Example Named Configuration Section
# Example 2: configuration forcing use of the Java Plugin in all cases (even if
# the client browser is on Windows)
[jpi]
baseHTMLJInitiator=basejpi.htm

# Example Named Configuration Section
# Example 3: 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 deploying WebUtil. Note that WebUtil is shipped with
# DS but not AS 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
baseHTMLjinitiator=webutiljini.htm
baseHTMLjpi=webutiljpi.htm
archive_jini=frmall_jinit.jar
archive=frmall.jar
lookAndFeel=oracle
```

C.2 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 Solaris

C.2.1 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 in the Windows registry
# will be used. If no value is found in the registry, the value used will
# be that defined in the environment in which the servlet engine (OC4J
# or JServ) was started.
#
# NOTES
# 1/ The Forms installation process should replace all occurrences of
#    <percent>FORMS_ORACLE_HOME<percent> with the correct ORACLE_HOME
#    setting, and all occurrences of <percent>O_JDK_HOME<percent> with
#    the location of the JDK (usually $ORACLE_HOME/jdk).
#    Please make these changes manually if not.
# 2/ Some of the variables below may need to be changed to suite your needs.
#    Please refer to the Forms documentation for details.
#
ORACLE_HOME=%FORMS_ORACLE_HOME%
```

```
#
# 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=%FORMS_ORACLE_HOME%\forms

# webutil config file path
WEBUTIL_CONFIG=%FORMS_ORACLE_HOME%\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).
# The Forms runtime executable and dll's are assumed to be in
# %ORACLE_HOME%\bin if they are not in the PATH.
# In addition, if you are running Graphics applications, you will need
# to append the following to the path (where <Graphics Oracle Home> should
# be replaced with the actual location of your Graphics 6i oracle_home):
#
# ;<Graphics Oracle Home>\bin;<Graphics Oracle Home>\jdk\bin
#

PATH=%FORMS_ORACLE_HOME%\bin;%FORMS_ORACLE_HOME%\jdk\jre\bin\client

#
# Settings for Graphics
# ------------------------------
# NOTE: These settings are only needed if Graphics applications
# are called from Forms applications. In addition, you will need to
# modify the PATH variable above as described above.
#

  #
  # Please uncomment the following and put the correct 6i
  # oracle_home value to use Graphics applications.
  #
  #ORACLE_GRAPHICS6I_HOME=<your Graphics 6i oracle_home here>

  #
  # Search path for Graphics applications
  #
  #GRAPHICS60_PATH=

#
# Settings for Forms tracing and logging
# ----------------------------------------------
# Note: This entry has to be uncommented to enable tracing and
# logging.

  #FORMS_TRACE_PATH=<FORMS_ORACLE_HOME>\forms\server


#
# System settings
# ---------------
# You should not normally need to modify these settings
```

```
  #
  FORMS=%FORMS_ORACLE_HOME%\forms

  #
  # Java class path
  #   This is required for the Forms debugger
  #   You can append your own Java code here)
  # frmsrv.jar, repository.jar and ldapjclnt10.jar are required for
  # the password expiry feature to work(#2213140).
  #
  CLASSPATH=%FORMS_ORACLE_HOME%\j2ee\OC4J_BI_Forms
\applications\formsapp\formsweb\WEB-INF\lib\frmsrv.jar;
%FORMS_ORACLE_HOME%\jlib\repository.jar;%FORMS_ORACLE_HOME%
\jlib\ldapjclnt10.jar;%FORMS_ORACLE_HOME%\jlib\debugger.jar;%FORMS_ORACLE_HOME%
\jlib\ewt3.jar;%FORMS_ORACLE_HOME%\jlib\share.jar;%FORMS_ORACLE_HOME%
\jlib\utj.jar;%FORMS_ORACLE_HOME%\jlib\zrclient.jar;%FORMS_ORACLE_HOME%
\reports\jlib\rwrun.jar;%FORMS_ORACLE_HOME%\forms\java\frmwebutil.jar
```

C.2.2 Default default.env File for Solaris

```
# default.env - default Forms environment file, Solaris 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 servlet engine (OC4J or JServ) was started.
#
# NOTES
# 1/ The Forms installation process should replace all occurrences of
#    <percent>FORMS_ORACLE_HOME<percent> with the correct ORACLE_HOME
#    setting, and all occurrences of <percent>O_JDK_HOME<percent> with
#    the location of the JDK (usually $ORACLE_HOME/jdk).
#    Please make these changes manually if not.
# 2/ Some of the variables below may need to be changed to suite your needs.
#    Please refer to the Forms documentation for details.
#
ORACLE_HOME=%FORMS_ORACLE_HOME%

  #
  # Search path for Forms applications (.fmx files, PL/SQL libraries)
  #
  FORMS_PATH=%FORMS_ORACLE_HOME%/forms

  # webutil config file path
  WEBUTIL_CONFIG=%FORMS_ORACLE_HOME%/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, repository.jar and ldapjclnt10.jar are required for
  # the password expiry feature to work(#2213140).
  #
  CLASSPATH=%FORMS_ORACLE_HOME%/j2ee/OC4J_BI_Forms
/applications/formsapp/formsweb/WEB-INF/lib/frmsrv.jar:%FORMS_ORACLE_HOME%
/jlib/repository.jar:%FORMS_ORACLE_HOME%/jlib/ldapjclnt10.jar:%FORMS_ORACLE_HOME%
/jlib/debugger.jar:%FORMS_ORACLE_HOME%/jlib/ewt3.jar:%FORMS_ORACLE_HOME%
```

```
/jlib/share.jar:%FORMS_ORACLE_HOME%/jlib/utj.jar:%FORMS_ORACLE_HOME%
/jlib/zrclient.jar:%FORMS_ORACLE_HOME%/reports/jlib/rwrun.jar:%FORMS_ORACLE_HOME%
/forms/java/frmwebutil.jar

  #
  # The PATH setting is not required for Forms if the Forms executables are
  # in <ORACLE_HOME>/bin. However, it is required if Graphics applications
  # are called from Forms applications.
  #
  PATH=%FORMS_ORACLE_HOME%/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 Graphics
  # --------------------------------
  # NOTE: These settings are only needed if Graphics applications
  # are called from Forms applications
  #

    #
    # Please uncomment the following and put the correct 6i
    # oracle_home value to use Graphics applications.
    #
    #ORACLE_GRAPHICS6I_HOME=<your Graphics 6i oracle_home here>

    #
    # Search path for Graphics applications
    #
    GRAPHICS60_PATH=

  #
  # Settings for Forms tracing and logging
  # ----------------------------------------------
  # Note: This entry has to be uncommented to enable tracing and
  # logging.

    #FORMS_TRACE_PATH=<FORMS_ORACLE_HOME>/forms/server



  #
  # 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=%FORMS_ORACLE_HOME%/lib32:%O_JDK_HOME%/jre/lib/sparc:
%O_JDK_HOME%/jre/lib/sparc/native_threads:%LD_LIBRARY_PATH%
```

C.3 base.htm, basejini.htm, and basejpi.htm Files

For a brief description and the locations of base.htm, basejini.htm, and basejpi.htm, see Chapter 3.2.1.3, "base.htm, basejini.htm, and basejpi.htm".

Four baseHTML files are created for your system by the Oracle Universal Installer during Oracle Application Server 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 Configuring Parameters with Application Server Control Console, 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.

> **Note:** baseHTML variables can be changed by modifying the corresponding parameter values in the Configuring Parameters with Application Server Control Console file.

The following baseHTML starter files are available in the `ORACLE_HOME/forms/server` directory:

- **basejini.htm**: This is a baseHTML file containing the tags required to run the Forms applet using Oracle JInitiator. It is suitable for browsers (only on Windows platforms) certified by Oracle to work in this manner (and which do not work using standard APPLET tags). See Default basejini.htm File for an example.
- **basejpi.htm**: This is the baseHTML file for Java Plug-in. The Forms Servlet uses this file if the client browser is not on Windows and the client browser is either Netscape or IE without the IE native setting.
- **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 whose native JVM is certified with Oracle Forms. See Default base.htm File for an example.

To create a new baseHTML file:

1. Place the new baseHTML file in any directory. Update the basejini.htm, basejpi.htm, or base.htm parameter in the formsweb.cfg file to contain the baseHTML file's full physical path location.
2. Copy the basejini.htm, basejpi.htm, or base.htm starter file, which is located in the `ORACLE_HOME/forms/server` directory.
3. Rename the file (for example, `order.htm)` .
4. Add or modify any text that is visible to the user (for example, text contained within <TITLE> and <BODY> tags).
5. 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 and baseHTMLJInitiator 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.

C.3.1 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 basejini.htm file, delete it from the file.

We recommend 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.3.1.1 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.3.2 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%

<!-- Forms applet definition (start) -->
<APPLET CODEBASE="%codebase%"
        CODE="oracle.forms.engine.Main"
        ARCHIVE="%archive%"
        WIDTH="%Width%"
        HEIGHT="%Height%">

<PARAM NAME="serverURL" VALUE="%serverURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs" VALUE="%escapeParams% module=%form% userid=%userid%
 sso_userid=%sso_userid%
 sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
 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%">

</APPLET>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

C.3.3 Default basejini.htm File

```
<HTML>
<!-- FILE: basejini.htm (Oracle Forms)                                  -->
<!--                                                                    -->
<!-- This is the default base HTML file for running a form on the       -->
<!-- web using JInitiator-style tags to include the 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           -->
<!-- baseHTMLJinitiator parameter in the Forms Servlet configuration -->
<!-- file (formsweb.cfg) to point to your new file instead of this.  -->

<HEAD><TITLE>%pageTitle%</TITLE></HEAD>

<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%

<!-- Forms applet definition (start) -->
<OBJECT classid="%jinit_classid%"
        codebase="/forms/jinitiator/%jinit_exename%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0">
<PARAM NAME="TYPE"       VALUE="%jinit_mimetype%">
<PARAM NAME="CODEBASE"   VALUE="%codebase%">
<PARAM NAME="CODE"       VALUE="oracle.forms.engine.Main" >
<PARAM NAME="ARCHIVE"    VALUE="%archive_jini%" >

<PARAM NAME="serverURL" VALUE="%serverURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs" VALUE="%escapeParams% module=%form%
userid=%userid% sso_userid=%sso_userid% sso_formsid=%sso_formsid%
sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
 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%">
<COMMENT>
<EMBED SRC="" PLUGINSPAGE="%jinit_download_page%"
        TYPE="%jinit_mimetype%"
        java_codebase="%codebase%"
        java_code="oracle.forms.engine.Main"
```

```
        java_archive="%archive_jini%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0"

        serverURL="%serverURL%"
        networkRetries="%networkRetries%"
        serverArgs="%escapeParams% module=%form% userid=%userid% sso_userid=
%sso_userid% sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
debug=%debug% host=%host% port=%port% %otherparams%"
        separateFrame="%separateFrame%"
        splashScreen="%splashScreen%"
        background="%background%"
        lookAndFeel="%lookAndFeel%"
        colorScheme="%colorScheme%"
        serverApp="%serverApp%"
        logo="%logo%"
        imageBase="%imageBase%"
        formsMessageListener="%formsMessageListener%"
        recordFileName="%recordFileName%"
        EndUserMonitoringEnabled="%EndUserMonitoringEnabled%"
        EndUserMonitoringURL="%EndUserMonitoringURL%"
        heartBeat="%heartBeat%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

C.3.4 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%

<!-- Forms applet definition (start) -->
<OBJECT classid="%jpi_classid%"
        codebase="%jpi_codebase%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0">
<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="%serverURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="%escapeParams% module=%form% userid=%userid% sso_userid=%sso_userid%
sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
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%">
<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"

        serverURL="%serverURL%"
        networkRetries="%networkRetries%"
        serverArgs="%escapeParams% module=%form% userid=%userid% sso_userid=
%sso_userid% sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
 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%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

C.4 web.xml

For a description and the location of web.xml, see Chapter 2, web.xml.

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 Developer Suite or OracleAS 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 OC4J). 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).

- Use a Forms Servlet configuration file other than the standard one (which is ORACLE_HOME/forms/server/formsweb.cfg).

  This can be done by uncommenting and changing the frmservlet's "configFileName" servlet parameter.

- 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).

C.4.1 Default web.xml File

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<!-- $Id: web.xml 29-apr-2004.13:43:19 ahousing Exp $
 Name
   web.xml
 Purpose
   Forms web application (WAR) configuration file
-->

<web-app>
   <display-name>Forms Services</display-name>
   <description>Oracle AS: Forms Services</description>

   <welcome-file-list>
```

```
      <welcome-file>lservlet</welcome-file>
   </welcome-file-list>

   <!-- Forms page generator servlet -->
   <servlet>
      <servlet-name>frmservlet</servlet-name>
      <servlet-class>oracle.forms.servlet.FormsServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
      <!-- During product installation the configFileName parameter is
           specified in the orion-web.xml file as a context parameter
           override (in iDS), or as a Java system property (in iAS).
           It is set to <oracle_home>/forms/server/formsweb.cfg.
           You can override that value here by editing and uncommenting the
           following servlet parameter setting: -->
      <!--
        <init-param>
           <param-name>configFileName</param-name>
           <param-value><your configuration file name goes here></param-value>
        </init-param>
      -->
      <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>
   </servlet>

   <!-- Forms listener servlet -->
   <servlet>
      <servlet-name>lservlet</servlet-name>
      <servlet-class>oracle.forms.servlet.ListenerServlet</servlet-class>
   </servlet>

   <!-- Forms servlet mappings. Allow these paths to the servlets:
           /forms/frmservlet or /forms/frmservlet/*: FormsServlet
           /forms/lservlet or /forms/lservlet/*: ListenerServlet
   -->
   <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>

   <!-- The following context parameter is only defined here so it can be
        overriden by the (site-specific) value in the orion-web.xml file.
    -->
    <context-param>
       <param-name>configFileName</param-name>
       <param-value></param-value>
    </context-param>
</web-app>
```

C.5 forms.conf

For a description and the location of forms.conf, see Chapter 2, forms.conf.

The following table describes the virtual paths and servlet mappings:

Table C–1 forms.conf Virtual Paths and Servlet Mappings

URL Path	Type	Maps to	Purpose
/forms/java	Alias	ORACLE_HOME/forms/java	codebase for Forms applet. Used to download the applet code to the user's web browser.
/forms/html	Alias	ORACLE_HOME/tools/frmweb/html	Access runform.htm (used to run any form for testing)
/forms/jinitiator	Alias	ORACLE_HOME/jinit	Oracle JInitiator download
/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 forms.conf

```
# Name
#   forms.conf
# Purpose
#   Apache mod_oc4j and mod_jserv configuration file for Forms Services.
#   This file should be included into the Oracle Apache HTTP Listener
#   configuration file (typically by adding an include statement to the
#   oracle_apache.conf file)
# Remarks
#   If Forms is to be used with JServ, the jserv.properties file needs editing
#   to add the "forms" servlet zone with properties file forms.properties
# Notes
#   Virtual paths: We use AliasMatch when defining virtual paths for
#   security reasons (prevents directory browsing).

  # Virtual path mapping for Forms Java jar and class files (codebase)
AliasMatch ^/forms/java/(..*) "%FORMS_ORACLE_HOME%/forms/java/$1"

  # Virtual path for JInitiator downloadable executable and download page
AliasMatch ^/forms/jinitiator/(..*) "%FORMS_ORACLE_HOME%/jinit/$1"

  # Virtual path for runform.htm (used to run a form for testing purposes)
AliasMatch ^/forms/html/(..*) "%FORMS_ORACLE_HOME%/tools/web/html/$1"

 # Virtual path for webutil
AliasMatch ^/forms/webutil/(..*) "%FORMS_ORACLE_HOME%/forms/webutil/$1"

# Configuration for JServ (if mod_jserv.c is available and not mod_oc4j.c)
<IfModule mod_jserv.c>
  # Only configure for JServ if mod_oc4j is NOT available:
  <IfModule !mod_oc4j.c>
    # Virtual path mapping for FormsServlet and ListenerServlet.
    # Purpose: paths to invoke the servlets should be /forms/frmservlet
    # and /forms/lservlet respectively.
```

```
    # We map frmservlet to servlet.frm, and lservlet to servlet.frml.
    # The apJServAction directives (below) will then remap those.
    AliasMatch ^/forms/frmservlet(.*) "/servlet.frm"
    AliasMatch ^/forms/lservlet(.*) "/servlet.frml"

    ApJServMount /forms/servlet /forms
    #
    # Let the servlets be called by file extension (e.g /servlet.frm)
    #
    ApJServAction .frm /forms/servlet/frmservlet
    ApJServAction .frml /forms/servlet/lservlet
    # Prevent access to the Forms Servlets by paths other than
    # /forms/frmservlet and /forms/lservlet.
    # 1. Prevent access via the .frm and .frml file extensions:
    <LocationMatch ^.*\.frm.*>
      order deny,allow
      deny from all
    </LocationMatch>
    # 2. Stop access by class (by paths like
    #    /forms/servlet/oracle.forms.servlet.FormsServlet)
    <LocationMatch ^/forms/servlet/oracle\.forms.*>
      order deny,allow
      deny from all
    </LocationMatch>
  </IfModule>
</IfModule>

# Config. for OC4J
<IfModule mod_oc4j.c>
    Oc4jMount /forms              OC4J_BI_Forms
    Oc4jMount /forms/frmservlet   OC4J_BI_Forms
    Oc4jMount /forms/frmservlet/* OC4J_BI_Forms
    Oc4jMount /forms/lservlet     OC4J_BI_Forms
    Oc4jMount /forms/lservlet/*   OC4J_BI_Forms
</IfModule>
```

C.6 Registry.dat

For a description and the location of Registry.dat, see Chapter 3.2.4.1, "Registry.dat".

The main reason you would want to edit this file is to change the icon settings (see Deploying Application Icons). You can also change the default font and font settings by changing the following section in the Registry.dat file:

```
default.fontMap.defaultFontname=Dialog
default.fontMap.defaultSize=900
default.fontMap.defaultStyle=PLAIN
default.fontMap.defaultWeight=PLAIN
```

Change any of the settings above to reflect your desired font setting. For example, if you want 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
```

Some fonts on Windows are not supported in Java. For this reason you can specify (map) Java-supported fonts that will 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. For more samples, see Default Registry.dat

C.6.1 Default 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
#
#
oracle.classById.1=oracle.forms.engine.Runform
oracle.classById.4=oracle.forms.handler.FormWindow
oracle.classById.5=oracle.forms.handler.AlertDialog
oracle.classById.6=oracle.forms.handler.DisplayList
oracle.classById.7=oracle.forms.handler.LogonDialog
oracle.classById.8=oracle.forms.handler.DisplayErrorDialog
oracle.classById.9=oracle.forms.handler.ListValuesDialog
oracle.classById.10=oracle.forms.handler.EditorDialog
oracle.classById.11=oracle.forms.handler.HelpDialog
oracle.classById.12=oracle.forms.handler.FormStatusBar
oracle.classById.13=oracle.forms.handler.MenuInfo
# oracle.classById.14=UNUSED
oracle.classById.15=oracle.forms.handler.ApplicationTimer
oracle.classById.16=oracle.forms.handler.MenuParametersDialog
oracle.classById.17=oracle.forms.handler.PromptListItem
oracle.classById.18=oracle.forms.handler.CancelQueryDialog
oracle.classById.257=oracle.forms.handler.TextFieldItem
oracle.classById.258=oracle.forms.handler.TextAreaItem
oracle.classById.259=oracle.forms.handler.FormCanvas
oracle.classById.261=oracle.forms.handler.ButtonItem
oracle.classById.262=oracle.forms.handler.CheckboxItem
oracle.classById.263=oracle.forms.handler.PopListItem
oracle.classById.264=oracle.forms.handler.TListItem
oracle.classById.265=oracle.forms.handler.CfmVBX
oracle.classById.266=oracle.forms.handler.CfmOLE
oracle.classById.267=oracle.forms.handler.RadioButtonItem
oracle.classById.268=oracle.forms.handler.ImageItem
oracle.classById.269=oracle.forms.handler.IconicButtonItem
oracle.classById.270=oracle.forms.handler.BlockScroller
oracle.classById.271=oracle.forms.handler.JavaContainer
oracle.classById.272=oracle.forms.handler.TabControl
oracle.classById.273=oracle.forms.handler.ComboBoxItem
oracle.classById.274=oracle.forms.handler.TreeItem
oracle.classById.281=oracle.forms.handler.PopupHelpItem

#
# 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,
Dialog,Dialog,Serif,Serif,Dialog,SansSerif

#
# The Application Level icon files are relative to the DOCUMENTBASE
#   example: icons/
# or an absolute URL.
#   example: http://www.forms.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.7 Default jvmcontroller.cfg

```
# Default JVM Controller
# This section defines the default values for jvm controllers
# under this Oracle Home. These values override the defaults
# for the dejvm executable.
[default]

# Example JVM Controller
# This section shows example values for a jvm controller. These
# value override 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.8 Default webutil.cfg

```
# ----------------------------------------------------------------------------
# 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. FileUpload 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
#
# 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.<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

# 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|94208|1.0|true
install.syslib.0.9.1=JNIsharedstubs.dll|65582|1.0|true
install.syslib.0.9.2=d2kwut60.dll|192512|1.0|true

# You can also add your own libraries in here, e.g.
#install.syslib.0.user.1=testwebutil.dll|204872|1.0|true




# 3. Upload / Download options
# 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 disable
#                                upload and download from the database server.
#  transfer.appsrv.enabled     : Can be TRUE or FALSE - allows you to 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
```

C.9 Default webutilbase.htm

```
<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) -->

<!-- Forms applet definition (start) -->
<APPLET CODEBASE="%codebase%"
        CODE="oracle.forms.engine.Main"
        ARCHIVE="%archive%,%webUtilArchive%"
        WIDTH="%Width%"
        HEIGHT="%Height%">

<PARAM NAME="serverURL" VALUE="%serverURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="%escapeParams% module=%form% userid=%userid% sso_userid=%sso_userid%
 sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
 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%">
<!-- 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.10 Default webutiljini.htm

```
<HTML>
<!-- FILE: webutiljini.htm (Oracle Forms)                          -->
<!--                                                               -->
<!-- This is the a HTML template file for running a form on the    -->
<!-- web using JInitiator-style tags to include the 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) -->
<OBJECT classid="%jinit_classid%"
        codebase="/forms/jinitiator/%jinit_exename%"
        WIDTH="0"
        HEIGHT="0"
        HSPACE="0"
        VSPACE="0">
<PARAM NAME="TYPE"       VALUE="%jinit_mimetype%">
<PARAM NAME="CODEBASE"   VALUE="%codebase%">
<PARAM NAME="CODE"       VALUE="oracle.forms.webutil.common.RegisterWebUtil" >
<PARAM NAME="ARCHIVE"    VALUE="%webUtilArchive%" >
<COMMENT>
<EMBED SRC="" PLUGINSPAGE="%jinit_download_page%"
        TYPE="%jinit_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) -->

<!-- Forms applet definition (start) -->
<OBJECT classid="%jinit_classid%"
        codebase="/forms/jinitiator/%jinit_exename%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0">
<PARAM NAME="TYPE"       VALUE="%jinit_mimetype%">
<PARAM NAME="CODEBASE"   VALUE="%codebase%">
<PARAM NAME="CODE"       VALUE="oracle.forms.engine.Main" >
<PARAM NAME="ARCHIVE"    VALUE="%archive_jini%,%webUtilArchive%" >

<PARAM NAME="serverURL" VALUE="%serverURL%">
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="%escapeParams% module=%form% userid=%userid% sso_userid=%sso_userid%
 sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
 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="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="%jinit_download_page%"
        TYPE="%jinit_mimetype%"
        java_codebase="%codebase%"
        java_code="oracle.forms.engine.Main"
        java_archive="%archive_jini%,%webUtilArchive%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0"

        serverURL="%serverURL%"
        networkRetries="%networkRetries%"
        serverArgs="%escapeParams% module=%form% userid=%userid% sso_userid=
%sso_userid% sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
 debug=%debug% host=%host% port=%port% %otherparams%"
```

```
        separateFrame="%separateFrame%"
        splashScreen="%splashScreen%"
        background="%background%"
        lookAndFeel="%lookAndFeel%"
        colorScheme="%colorScheme%"
        serverApp="%serverApp%"
        logo="%logo%"
        imageBase="%imageBase%"
        formsMessageListener="%formsMessageListener%"
        recordFileName="%recordFileName%"
        EndUserMonitoringEnabled="%EndUserMonitoringEnabled%"
        EndUserMonitoringURL="%EndUserMonitoringURL%"
        heartBeat="%heartBeat%"
        WebUtilLogging="%WebUtilLogging%"
        WebUtilLoggingDetail="%WebUtilLoggingDetail%"
        WebUtilErrormode="%WebUtilErrorMode%"
        WebUtilDispatchMonitorInterval="%WebUtilDispatchMonitorInterval%"
        WebUtilTrustInternal="%WebUtilTrustInternal%"
        WebUtilMaxTransferSize="%WebUtilMaxTransferSize%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

C.11 Default webutiljpi.htm

```
<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) -->

<!-- Forms applet definition (start) -->
<OBJECT classid="%jpi_classid%"
        codebase="%jpi_codebase%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0">
<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="%serverURL%">
```

```
<PARAM NAME="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs" VALUE="%escapeParams% module=%form% userid=%userid%
 sso_userid=%sso_userid% sso_formsid=%sso_formsid% sso_subDN=%sso_subDN%
sso_usrDN=%sso_usrDN% 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="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"

        serverURL="%serverURL%"
        networkRetries="%networkRetries%"
        serverArgs="%escapeParams% module=%form% userid=%userid% sso_userid=
%sso_userid% sso_formsid=%sso_formsid% sso_subDN=%sso_subDN% sso_usrDN=%sso_usrDN%
 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%"
        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

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z