Oracle® Application Server Forms Services

Deployment Guide

10*g* (9.0.4) for Windows and UNIX

Part No. B10470-02

March 2004

ORACLE®

Oracle Application Server Forms Services Deployment Guide 10*g* (9.0.4) for Windows and UNIX

Part No. B10470-02

Copyright © 2004 Oracle. All rights reserved.

Primary Author: Orlando Cordero

Contributor: Emerson deLaubenfels, Art Housinger, Srini Indla, David Klein, Phil Kuhn, Chris Lewis, Chris Lowes, Duncan Mills, Frank Nimphius, Stephen Noton, Hiromichi Nozaki, Robin Zimmermann.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

3 Basics of Deploying Oracle Forms Applications

4 Configuring Forms Services with Enterprise Manager

5 Using OracleAS Forms Services with the HTTP Listener and OC4J

6 Using Forms Services with Oracle Application Server Single Sign-On and OID

7 Tracing and Diagnostics

8 Performance Tuning Considerations

A JInitiator

B Sample Configuration Files

Send Us Your Comments

Oracle Application Server Forms Services Deployment Guide 10*g* (9.0.4) for Windows and UNIX

Part No. B10470-02

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com: Attention Forms Documentation
- Postal service:

  Oracle Corporation
  Oracle Forms
  200 Oracle Parkway
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

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. 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 additional information, visit the Oracle Accessibility Program Web site at

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

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, 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, JAWS 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.

Structure

This manual contains the following chapters and appendices:

This sample manual contains one part, two chapters, and one appendixes. (Insert this chapter, appendix, and parts as cross-references so that the links are apparent in HTML.)

Chapter 1, "Introduction"

Introduces you to deploying applications on the Oracle Internet Platform and provides an overview of the Forms Services architecture and components.

Chapter 2, "Forms Services Security Overview"

Provides overview information about using the security features of Oracle Application Server Forms Services

Chapter 3, "Basics of Deploying Oracle Forms Applications"

Introduces you to the basic files you need to configure OracleAS Forms Services and describes the steps for deploying applications.

Chapter 4, "Configuring Forms Services with Enterprise Manager"

Describes advanced topics in configuring OracleAS Forms Services and provides guidelines and tips for designing Oracle Forms applications for Web deployment.

Chapter 5, "Using OracleAS Forms Services with the HTTP Listener and OC4J"

Describes how OracleAS Forms Services works using Oracle HTTP Listener and OC4J.

Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On and OID"

Describes using OracleAS Forms Services with single sign-on and Oracle Internet Directory.

Chapter 7, "Tracing and Diagnostics"

Describes the Oracle Enterprise Manager (OEM) system management tool.

Chapter 8, "Performance Tuning Considerations"

Describes tracing and diagnostic tools that are available with Forms allow you to analyze the performance and resource consumption of your Oracle Forms applications at runtime.

Chapter A, "JInitiator"

Describes the benefits of using Oracle JInitiator and how to set up the Oracle JInitiator plug-in.

Chapter B, "Sample Configuration Files"

Sample environment, parameter, base.htm, basejini.htm, web.xml, forms90.conf, baseie.htm files.

Related Documents

For more information, see the following manuals in the Oracle Other Product One Release 7.0 documentation set or in the Oracle Other Product Two Release 6.1 documentation set:

- *Oracle Application Server Release Notes*
- *Oracle Developer Suite Release Notes*
- Oracle Forms Migrating Forms Applications from Forms6i (Part No. B10469-02)
- Oracle Forms Developer Online Help, available from the Help menu in Forms Developer.

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

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. For a description of each chapter in this guide see the Preface.

This chapter contains the following sections:

1.1 The Oracle Internet Platform

With Oracle9*i database* to manage data, Oracle Developer Suite (OracleDS) 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:

- Oracle Application Server (OracleAS)
- Oracle Developer Suite (OracleDS)
- Oracle9i Database

1.1.1 Oracle Application Server (OracleAS)

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 (OracleDS)

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

OracleDS 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 the *Oracle9i* Database Server.

1.1.3 Oracle9*i* Database

Oracle9i Database Server is the latest generation of the world's most popular RDBMS. Among the numerous new capabilities are unlimited scalability and industry-leading reliability with Oracle9*i* 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 *Oracle9i* Database Server and applications built with Oracle development tools. By leveraging a common technology stack, Oracle Application Server can transparently scale an *Oracle9i* Database Server 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 *Oracle9i* Database Server. A Java applet provides the client user interface.

New features for Forms Services include:

- Runtime Pooling (see Chapter 8.1.2, "Forms Services Web Runtime Pooling")
- Improved Enterprise Manager Web interface (see Chapter 6.3, "Enabling Single Sign-On for an Application"
- Single Sign-on (SSO) improvements (see Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On and OID")
- Support for deployment on the Web using the Forms Listener Servlet.

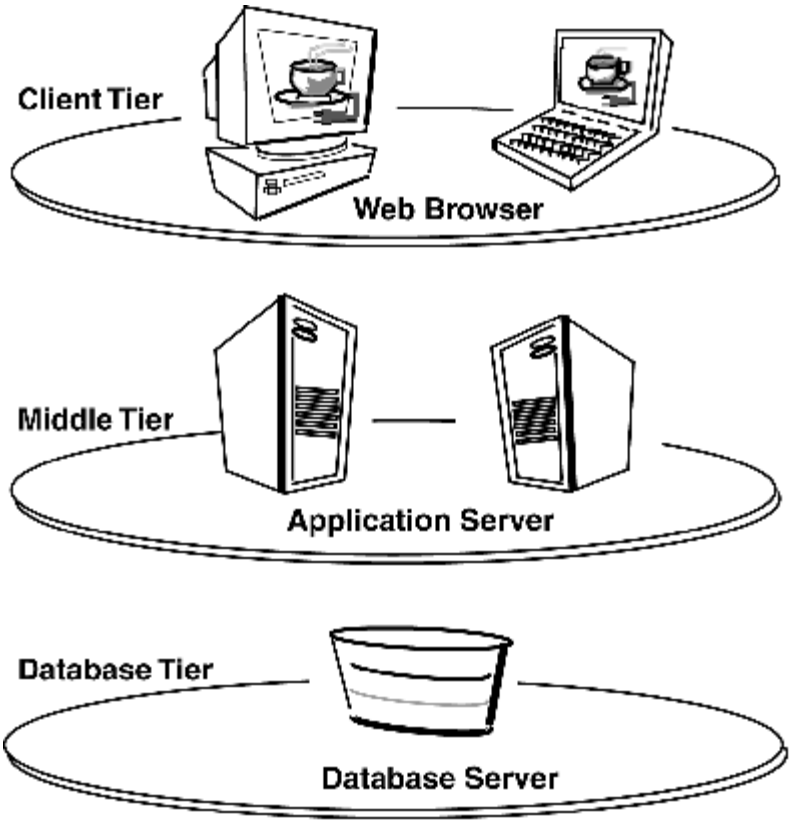
- Integration with OC4J (see Chapter 5, "Using OracleAS Forms Services with the HTTP Listener and OC4J")
- Integration with OID (see Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On and OID")
- Integration with Enterprise Manager for easier administration and manageability (see Chapter 7, "Tracing and Diagnostics")
- Tracing and logging improvements (see Chapter 8, "Performance Tuning Considerations")

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



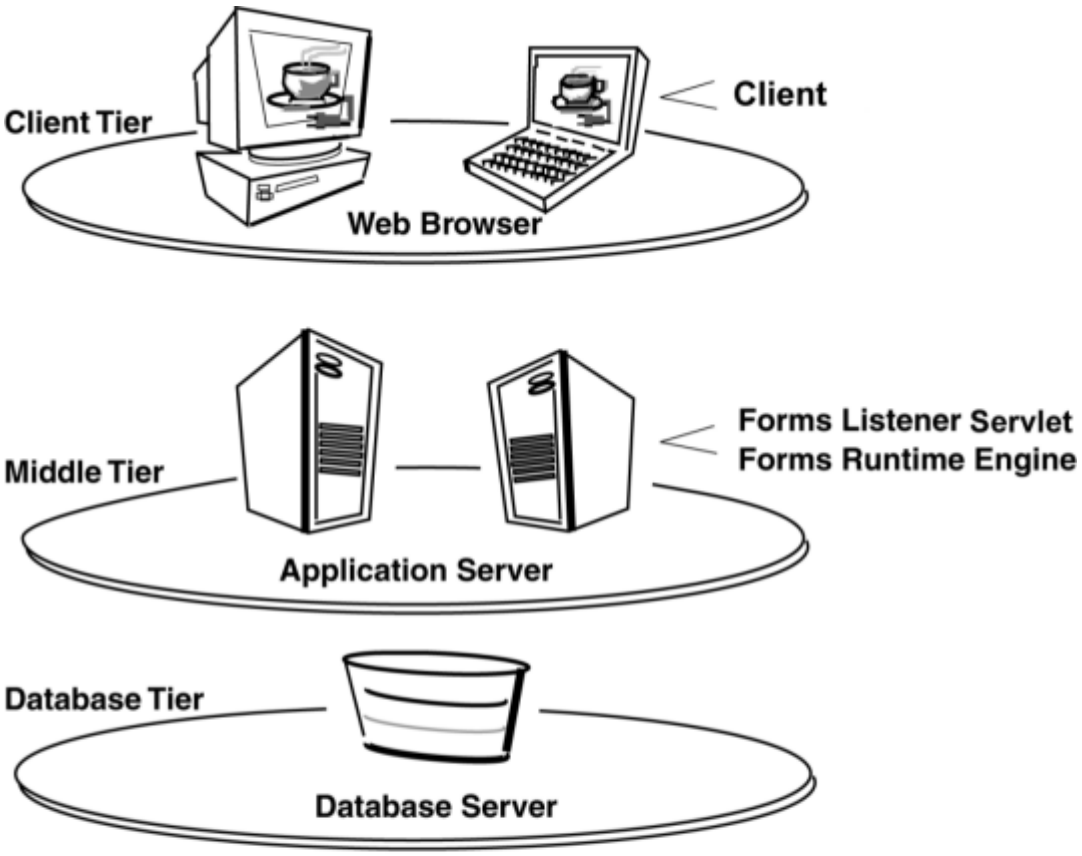
1.4 OracleAS Forms Services Components

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.

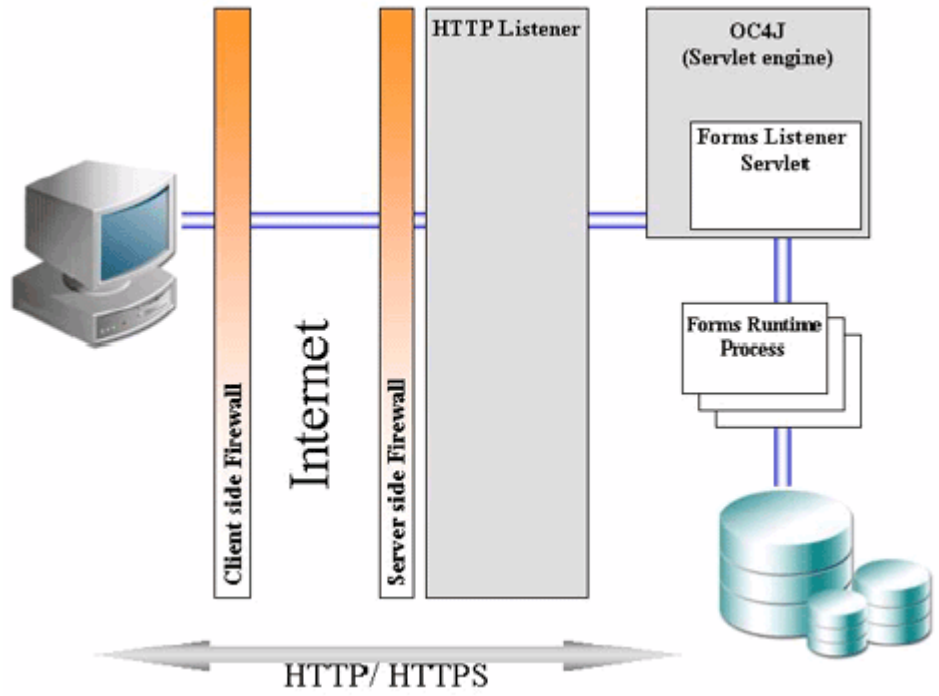
1.5 Forms Listener Servlet

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 machines 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:

- About OracleAS Forms Services Security
- Configuring OracleAS Forms Services Security

See Also: For additional information about security, refer to the following documents:

- The *Oracle Application Server 10g 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 (OID) 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 single sign-on environment, the Forms servlet queries OID for the value of the combined unique key that is constructed from the user's 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 OID 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 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 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 (DAS). DAS is a Web-based user interface for SSO users and delegated administrators to administer self-service data in OID for which they are authorized.

Once a user account is created in OID, 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 DAS. 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 SSO 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 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 single sign-on protected Forms applications as well as non 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 single sign-on parameter in the application definition of the `forms90/server/formsweb.cfg` configuration file.

2.1.3.2 Dynamic Resource Creation in OID

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 OID 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 OID 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 SSO 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 Chapter 6.5, "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 single sign-on for your Forms applications, Oracle Application Server Forms Services handles much of the configuration and interaction with OID. For more information about configuring single sign-on and OID, see Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On and OID".

2.2 Configuring OracleAS Forms Services Security

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

2.2.1 Configuring Oracle Identity Management Options for Oracle Forms

OracleAS Forms Services can be configured to create resources dynamically in OID, or have a user with no OID resource use a common resource.

For more information, see Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On and OID".

2.2.2 Configuring Oracle Forms Options for Oracle9*i*AS Security Framework

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

Chapter 4, "Configuring Forms Services with Enterprise Manager"

Chapter 5, "Using OracleAS Forms Services with the HTTP Listener and OC4J"

Chapter 6, "Using Forms Services with Oracle Application Server Single Sign-On and OID"

Chapter 7, "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 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 with Enterprise Manager".

This section contains the following sub-sections:

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

3.1.1 Oracle Forms Configuration Files

Oracle Forms configuration files allow you to specify parameters for your Forms, which you manage through the Application Server Control. This section contains the following sub-sections:

- base.htm, basejini.htm, basejpi.htm, and baseie.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.

3.1.1.1 default.env

Location: forms90/server

This file contains environment settings for Forms runtime and can be found in the <ORACLE_HOME>/forms90/server directory. On UNIX, default.env should include the PATH and LD_LIBRARY_PATH.

3.1.1.2 formsweb.cfg

Location: *forms90/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.

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

3.1.1.3 base.htm, basejini.htm, basejpi.htm, and baseie.htm

Location: *forms90/server.*

The baseHTML files (base.htm, basejini.htm, basejpi.htm, and baseie.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 B.3, "base.htm, basejini.htm, basejpi.htm, and baseie.htm Files".

3.1.1.4 ftrace.cfg

Location: *forms90/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.

For more information about ftrace.cfg, see Chapter 7, "Tracing and Diagnostics".

3.1.2 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 forms90app.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

3.1.2.1 web.xml

Location: *j2ee/OC4J_BI_FORMS/applications/forms90app/forms90web/WEB-INF/web.xml*.

Once Oracle Application Server Forms Services has been installed and configured, the web.xml file is located in the directory j2ee/OC4J_BI_FORMS/applications/forms90app/forms90web/WEB-INF underneath <ORACLE_HOME>. It defines the aliases `f90servlet` and `l90servlet` for the Forms Servlet and the Forms Listener Servlet.

For more information about web.xml, see Chapter B.4, "web.xml".

3.1.2.2 Directory structure for Oracle Forms OC4J files

During Oracle Application Server installation and configuration, the Forms EAR file (forms90app.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/forms90app
+META-INF
-application.xml (defines the structure of the ear file)
+forms90web
+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
       -f90srv.jar (contains the Forms Servlet and Listener Servlet code)
```

3.1.3 Oracle HTTP Listener Configuration Files

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

3.1.3.1 forms90.conf

Location: *forms90/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). Forms90.conf defines virtual directors (aliases) and servlet mount points to map URL requests to the Forms Servlets running in the OC4J servlet engine.

For more information about forms90.conf, see Chapter B.5, "forms90.conf".

3.1.4 Standard Fonts and Icons File

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

3.1.4.1 Registry.dat

Location: *forms90/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 Chapter B.6, "Registry.dat".

3.2 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 (ifweb90.exe on Windows or f90webm on Solaris) for each new Oracle Application Server Forms Services session.

For more information about how Forms Services run, see OracleAS Forms Services in Action.

3.2.1 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.

Create a configuration section in the Forms Web Configuration page of Oracle Enterprise Manager Application Server Control so that Oracle Application Server Forms Services can access your application module.
The following table 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.

1. Make sure the .fmx file location is specified in the FORMS90_PATH environment variable. For example, if your .fmx file is located in `d:\my_files\applications`, in the `FORMS90_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.

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:

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.

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

   `http://mymachine.com:7777/forms90/f90servlet?`

   where "mymachine" is the name of your machine and "7777" is the port used by your HTTP Listener.

   Since you create 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://mymachine.com:7777/forms90/f90servlet?config=applicati
   on
   ```

3.2.2 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.

  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 "Creating configuration sections in Enterprise Manager"). For example, in the configuration section you create for `my_application`, you can add or change these parameters and their values:

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

Parameter Name	Parameter Value
baseHTML	mybase.htm
baseHTMLjinitiator	mybasejini.htm
baseHTMLjpi	mybasejpi.htm
baseHTMLie	mybaseie.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.

3.2.3 Creating configuration sections in Enterprise Manager

Under the configuration sections you created in step 2 of 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 "Default Forms Configuration Parameters".

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

3.2.3.1 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://mymachine.com:7777/forms90/f90servlet?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://mymachine.com:7777/forms90/f90servlet?config=ienative&form=hrapp
```

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

> **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.

3.2.4 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.

3.2.4.1 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; 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).

3.2.4.2 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.

3.2.4.3 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`.

3.2.4.4 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.

3.2.4.5 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">
```

3.3 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 machine called "mymachine.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://mymachine.com:7777/forms90/f90servlet?config=ienative&form=hrapp`
   In this case, the (top level) form module to be run is called "hrapp" using the configuration section called "ienative"

2. Oracle HTTP Server listener receives the request. It forwards the request to OC4J, since the path `/forms90/f90servlet` matches one of the OC4J mount directives in the forms90.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 `/forms90`). It maps the request to the Forms Servlet (using the f90servlet servlet 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>/forms90/server/formsweb.cfg) is used.
- Determines which configuration section to use in the formsweb.cfg file. Since the URL contains the query parameter "config=ienative", the [ienative] 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 machine, they are prompted to install the plug-in. In the case of JInitiator, the download location is under the virtual path /forms90/jinitiator (a virtual path defined in the forms90.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://mymachine.com:7777/forms90/java/f90all_jinit.jar

   The virtual path definition in the forms90.conf file for "/forms90/java" allows the applet code to be loaded from the Web server.

   Note: The Forms applet code (for example, f90all_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://mymachine.com:7777/forms90/l90servlet`.
5. The Oracle HTTP Server listener receives the request. It forwards the request to OC4J, since the path "/forms90/l90servlet" matches one of the OC4J mount directives in the forms90.conf file (the one for the Forms Listener Servlet).
6. The Forms Listener Servlet (l90servlet) starts up a Forms runtime process (ifweb90.exe or f90webm) 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), or in one of the directories named in the FORMS90_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.

3.4 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 (f90all.jar and f90all_jinit.jar) that group and zip classes together for efficient delivery across the network to the client. f90all_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 A, "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 the following table:

Table 3–5 Web Browsers and the appropriate baseHTML file for each

Browser detected	IE parameter setting	Base HTML file used
Internet Explorer 5.x or 6*	native VM**	baseie.htm
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 with Enterprise Manager

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 Application Server Control as your needs change.

4.2 Enterprise Manager and Oracle Forms

The Enterprise Manager Application Server Control 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://<machine.domain>:1810
```

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 to:

- Monitor metrics for an Forms Services instance. See Chapter 8.1.1.1, "Monitoring Forms Services Instances" for more information.
- Monitor metrics for user sessions See Chapter 8.1.1.3, "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" for more information.
- Configure Forms Trace and monitor trace metrics. See Chapter 7.1.1, "Configuring Forms Trace" and Chapter 7.1.6, "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 8.1.3, "Forms Services Utilities" and Chapter 8.2, "Tuning OracleAS Forms Services Applications" for more information

4.2.1 Using Enterprise Manager to Manage Forms Sessions

By default, Enterprise Manager 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 the following variable is set in the default section. You can do this either through EM, or manually on the server.

   ```
   em_mode=1
   ```

   This will let EM 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 EM, or manually on the server.

   ```
   allow_debug=true
   ```

   This will let you turn tracing on and off within EM.

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 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 EM.

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 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

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

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

1. Using Application Server Control, 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, see "Introduction to Administration Tools" in the *Oracle Application Server 10g 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.

4.3 Configuring Forms Services

Use the Configuration page in Application Server Control 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 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.
2. From the Application Server Control 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 and click **Go**.
   - 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 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

For a description and the location of the Forms Servlet configuration file (formsweb.cfg), see Chapter 3.1.1.2, "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:

Table 4–1 Default Configuration Parameters that Specify Files

Parameter	Value
baseHTML	base.htm
`baseHTMLie`	baseie.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>/forms90/server`.

4.3.2 Managing Configuration Sections

You create new configuration sections from the **Configuration** tab of Application Server Control, 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.
2. From the Application Server Control 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:

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

Parameter	Value
forms	<module>
separateFrame	True

Table 4–2 (Cont.) Sample Parameters to Add to a New Configuration Section

Parameter	Value
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/forms90/f90servlet?config=sepwin
```

You can also use Application Server Control to create named configuration sections (see Chapter 7, "Tracing and Diagnostics").

See Appendix B, "Sample Configuration Files" 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, select the radio button next to the section name and click **Edit**.

To delete a named configuration:

1. Start the Enterprise Manager Application Server Control.
2. From the Application Server Control 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 returns to the Forms Configuration tab and displays the remaining configurations at the bottom of the page.

4.3.3 Managing Parameters

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

To edit a parameter in a configuration section:

1. From the Configuration tab of Enterprise Manager Application Server Control, select the radio button next to the configuration section to which you want to edit a parameter.
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, 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 Forms Configuration Section Parameters 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 Confirm 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 single sign-on parameters, see Chapter 6.3, "Enabling Single Sign-On for an Application".

These sections include:

- System Default Configuration Parameters
- Runform parameters (serverArgs parameters)
- HTML page title, attributes for the BODY tag and HTML to add before and after the form
- Applet or Object Parameters
- Parameters for JInitiator
- Parameters for Sun's Java Plug-in
- Enterprise Manager Configuration Parameters
- OID (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.

Table 4–3 System Default Configuration Parameters

Parameter	Required / Optional	Parameter Value
`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.
`baseHTMLie`	optional	Physical path to the HTML file that contains Internet Explorer 5 tags, for example the CABBASE tag. The default path is `<ORACLE_HOME>/forms90/server/baseie.htm`. This is required when using the Internet Explorer native JVM.
`HTMLdelimiter`	required	Delimiter for variable names. Defaults to %.
`workingDirectory`	required	Defaults to `<ORACLE_HOME>/forms90` if not set.
`envFile`	required	This is set to default.env in the formsweb.cfg file.

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

Parameter	Required / Optional	Parameter Value
`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 configuration section is searched because the name is not yet known.
`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`.

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 3.2.4, "Specifying Special Characters in Values of Runform Parameters" for information about how runform handles certain special characters that are specified in runform parameter values.

Table 4–4 Runform Parameters (serverArgs Parameters)

Parameter	Required / Optional	Parameter Value
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 FORMS90_TIMEOUT, the user's session will be kept alive, even if they are not actively using the form.
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 3.2.4, "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 forms90/f90servlet/admin screen are activated. forms90/f90servlet/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 Forms EM User Sessions screen. The default value is false; the test.fmx application is executed if an administrative function is attempted.
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.

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

Parameter	Required / Optional	Parameter Value
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.
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 may not be set using URL query parameters.

Table 4–5 HTML Page Parameters

Parameter	Required / Optional	Parameter Value
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

All of the following 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
serverURL	required	/forms90/l90servlet (see Chapter 1, Forms Listener Servlet)
codebase	required	Virtual directory you defined to point to the physical directory <ORACLE_HOME>/forms90/java, where, by default, the applet JAR files are downloaded from. The default value is /forms90/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
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>/forms90/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 f90all.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 f90all_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 f90all.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

Table 4–7 Parameters for JInitiator

Parameter	Required / Optional	Parameter Value
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 /forms90/jinitiator/us/JInitiator/jinit.download.htm.
jinit_classid	required (IE only)	Default is clsid:CAFECAFE-0013-0001-0009-ABCDEFABCDEF
jinit_exename	required	Default is jinit.exe#Version=1.3.1.9
jinit_mimetype	required (Netscape only)	Default is application/x-jinit-applet;version=1.3.1.9
baseHTMLJInitiator	required	Physical path to HTML file that contains JInitiator tags.

4.3.4.6 Parameters for Sun's Java Plug-in

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

Parameter	Required / Optional	Parameter Value
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

Table 4–9 Enterprise Manager Configuration Parameters

Parameter	Required / Optional	Parameter Value
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.

4.3.4.8 OID (Oracle Internet Directory) Configuration Parameters

Table 4–10 Enterprise Manager Configuration Parameters

Parameter	Required / Optional	Parameter Value
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 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 FORMS90_PATH for the Forms runtime executable (ifweb90.exe on Windows and f90webm on UNIX) 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>/forms90/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. Environment files must be created manually in `<ORACLE_HOME>/forms90/server` with a `.env` extension.

Likewise, environment files cannot be deleted from EM. For a new environment file to be picked up by Application Server Control 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.
`FORM90_PATH`	`<ORACLE_HOME>\forms90` (default)	Specifies the path that Oracle Forms searches when looking for a form, menu, or library to run. For **Windows**, separate paths with a *semi-colon* (;). For **UNIX**, separate paths with a *colon* (:).
`FORMS90_TIMEOUT`	Default: 15 Valid Values: 3 – 1440 (1 day) Example: `FORMS90_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.

Table 4–11 (Cont.) Default Environment Variables

Environment Variable	Valid Values	Purpose
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
GRAPHICS60_PATH	ORACLE_GRAPHICS6I_HOME=<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:

- Allowing New Users Sessions
- Disabling New User Sessions
- Terminating a User Session on a Forms Services Instance

4.5.1 Allowing New Users Sessions

By default, users can create new Forms sessions, which is indicated by the green traffic light. You can also enable users to create Forms sessions after you've enabled 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 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 Application Server Control.
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. Provide the credentials in the dialog box that displays (the user name and password that is required is the same one that was used when Forms Services was installed).
7. Click **OK**.

   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.

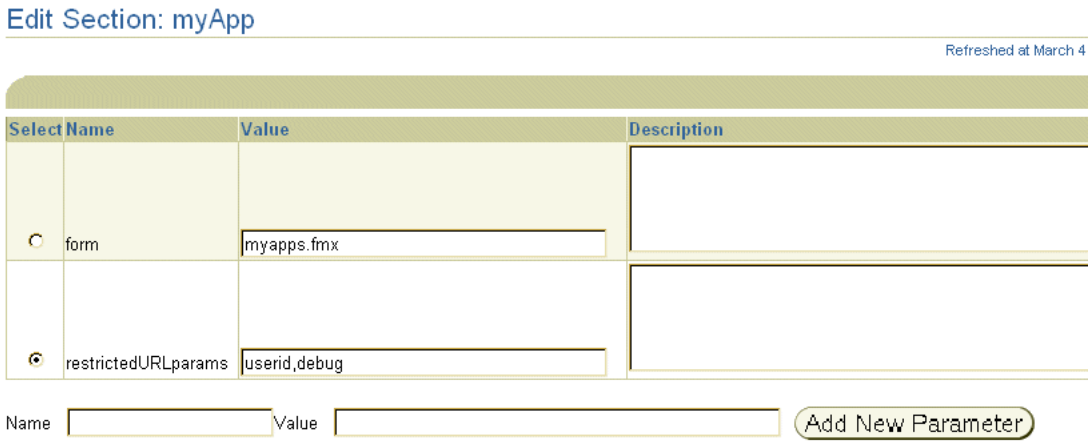
A Forms administrator can override this default behavior, and give the Forms administrator full control over what parameter can be used in the URL.

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 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 Sample Configuration Section for myApp



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.

4.7 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 3.2.4, "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://otn.oracle.com/products/forms/.*

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. Additional information can be found at http://otn.oracle.com/products/forms.

4.9.1 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 `forms90` 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 `forms90/myapp`.

To change the default location, you can set the imageBase parameter to codebase in the Forms Web Configuration page of Enterprise Manager Application Server Control. Alternatively, you can change the default.icons.iconpath value of the Registry.dat file in the `forms90/java/oracle/forms/registry` directory.

Setting the imageBase parameter to codebase enables Oracle Forms to search the `forms90/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.

4.9.1.1 Storing Icons in a Java Archive

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 myjar.jar *.gif zips` 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 `forms90/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=f90all_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 f90all.jar file,

4.9.1.2 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>/forms90/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)

1. 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

Icon Location	When	How
DocumentBase	Default. Applications with few or no custom icons.	Store icons in forms90 directory or in a directory relative to forms90.
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.

4.9.2 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.

4.9.3 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 roundtrips 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 roundtrip is necessary to download the Jar file.

4.9.3.1 Creating a Jar File

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://www.javasoft.com.

For example:

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

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

4.9.3.2 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=f90all.jar, icons.jar
imageBase=codebase
```

4.9.4 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 Enterprise manager 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 host 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.

4.9.4.1 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`	`http://host/documentbase/mydir`
(specified in your application file)	(relative path)

Table 4–13 (Cont.) Search Paths for Icons

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

Table 4–14 Search Paths for Images

file.gif (specified, for example, in formsweb.cfg as `splashscreen=file.gif`)	`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`

4.9.4.2 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)

Table 4–16 (Cont.) Image Search Paths Used by Forms Services

Location Specified	Search Path Used by Forms Services
`/mydir/file.gif`	`http://host/mydir/file.gif`
(specified in your HTML file)	(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 we 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 How Language Detection Works

When the Forms Servlet receives a request for a particular configuration (for example, `http://myserv/servlet/f90servlet?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/f90servlet`), it will look for a language-specific section in the default section matching the first language (for example, [.fr]).

4.10.2.1 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 FORMS90_PATH (to pick up language-specific fmx files). Using different `workingDirectory` settings provides another way to pick up language-specific.fmx files.

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.

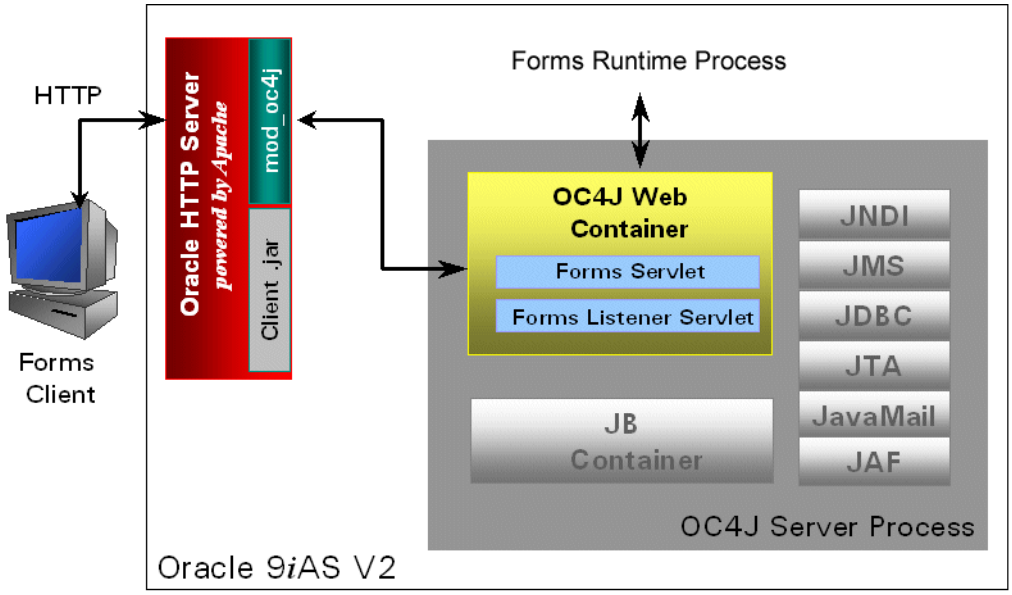
This chapter contains the following sections:

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 Performance/Scalability Tuning).

OC4J runs using the following 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 OracleAS Disk 1 CD or OTN at http://otn.oracle.com/docs/products/ias/content.html).

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 OracleAS Disk 1 CD or OTN at http://otn.oracle.com/docs/products/ias/content.html).

For more Forms-specific information, see the *OracleDS Forms Developer and OracleAS Forms Services Release Notes*.

The following images illustrate four possible deployment scenarios:

- Balancing incoming requests between multiple OC4J engines on the same host as the Oracle HTTP Listener.
- Balancing incoming requests between multiple OC4J engines on a different host to the Oracle HTTP Listener.
- Balancing incoming requests between multiple OC4J engines on multiple different hosts and multiple different hosts each running an Oracle HTTP Listener.
- 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 Case 1: Multiple OC4J engines on the same host as the Oracle HTTP Listener

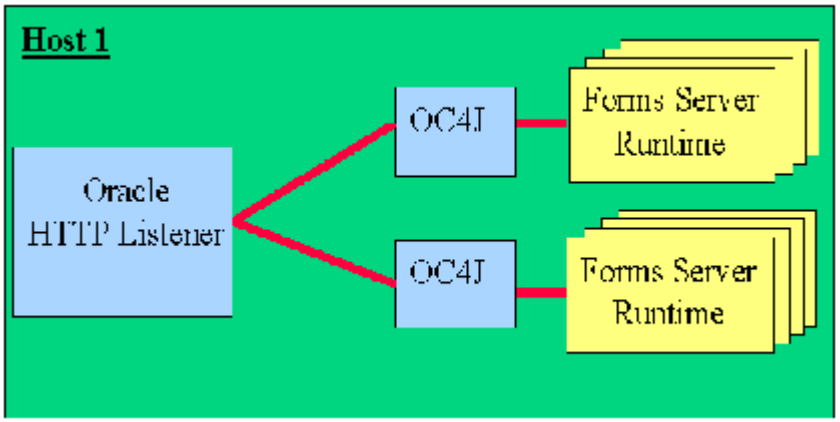


Figure 5–3 Case 2: Multiple OC4J engines on a different host to the Oracle HTTP Listener

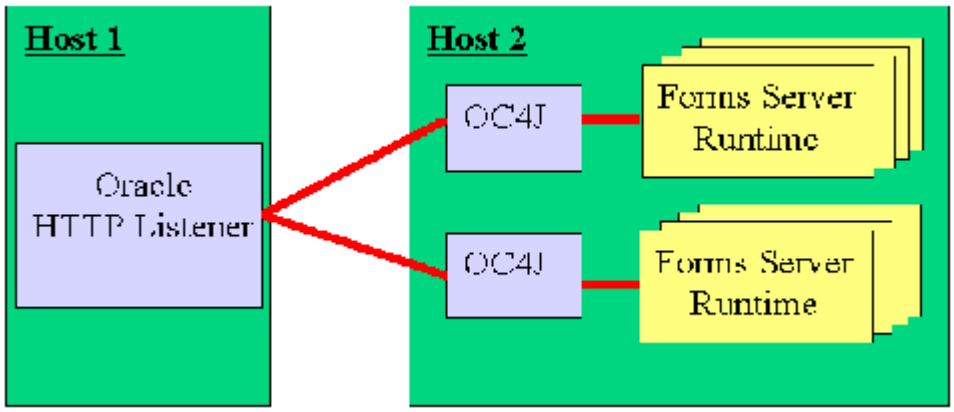


Figure 5–4 Case 3: Multiple OC4J engines and multiple Oracle HTTP Listeners on different hosts

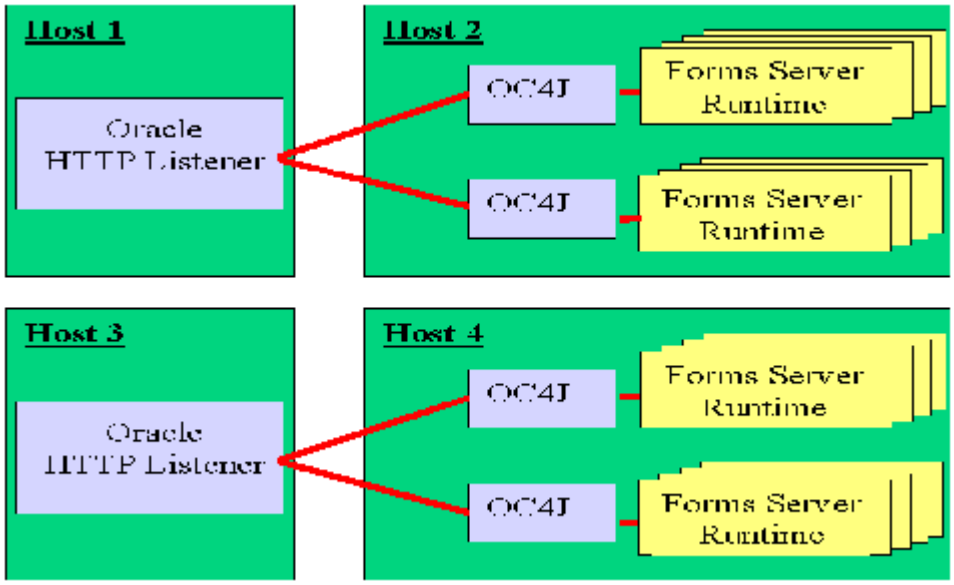
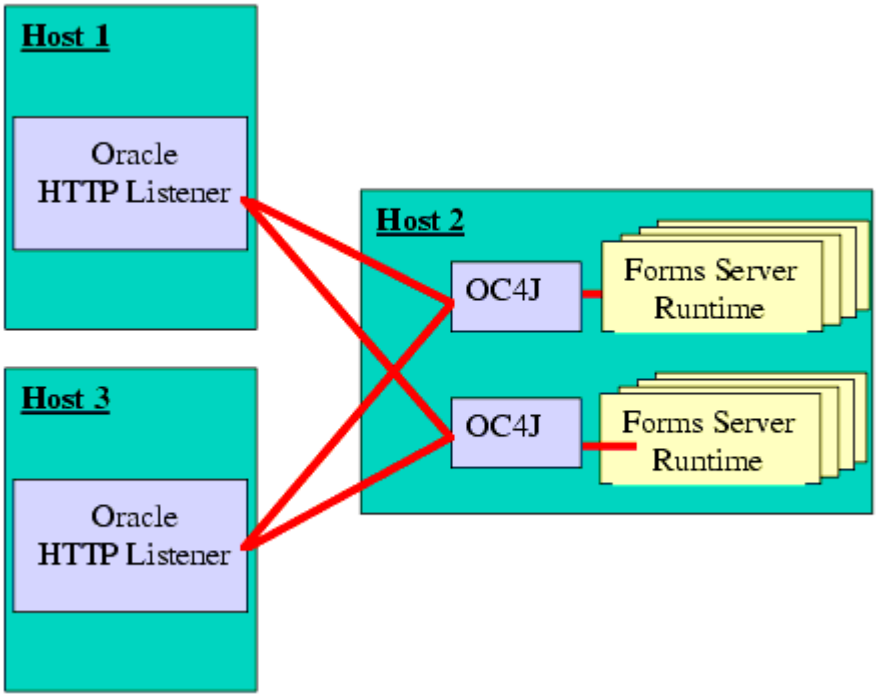


Figure 5–5 Case 4: Multiple Oracle HTTP Listeners on different hosts with multiple OC4J engines on one host



For more information about tuning and optimizing Forms Services with the HTTP Listener and OC4J, see Oracle Application Server Performance Guide (available on OracleAS Disk 1 CD or OTN at http://otn.oracle.com/docs/products/ias/content.html).

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 machine 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 A, "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 `/forms90`, such as `https://myserver.com/forms90/f90servlet?config=myApp`.

The codebase and server URL values set up by the Oracle Application Server installation process are `/forms90/java` and `/forms90/l90servlet`. As these are under the document base of the page (/forms90), authenticating proxies will work.

6

Using Forms Services with Oracle Application Server Single Sign-On and OID

Single sign-on is the ability of 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 be run in a single sign-on (SSO) environment using Oracle Single Sign-On Server and Oracle Internet Directory (OID) to store user name and password information. SSO is designed to work in Web environments where multiple Web-based applications are accessible from a Browser. Without SSO, 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.

The Oracle Single Sign-On Server can be used to enable 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 single sign-on architecture based on Oracle Single Sign-On Server and the Oracle Internet Directory (OID). Oracle Application Server Forms Services, a component of the Oracle Application Server, 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.

This chapter contains the following sections:

6.1 What's New with SSO and OID and Forms

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

- Dynamic Resource Creation When A Resource Is Not Found In OID
- Support for Default Preferences in OID to Define Forms Resources
- Support for Dynamic Directives With Forms SSO
- Support for Database Password Expiration for Forms Running with Single Sign-On

6.1.1 Dynamic Resource Creation When A Resource Is Not Found In OID

A user connects to Forms and is authenticated by mod_osso in combination with the SSO Server and OID. Once the user is authenticated, the user is directed to the Forms Servlet which takes the user's request information containing the SSO user name. The user name and the application name build a unique pair that identifies the user's resource information for this application in OID.

When an authenticated Forms user has no resource information for the particular application that is being requested in OID, then the user receives a Forms error message saying that no resource is available for this application, by default.

The way Forms Services handles the missing resource information is customizable by the application or Forms Services administrator. The following options are available:

- Display Forms error message (default)
- Redirect the user to a defined URL
- Display the Forms logon screen

The redirection URL is provided by the system administrator in the Forms configuration files and should be either absolute or relative.

6.1.2 Support for Default Preferences in OID to Define Forms Resources

In previous releases, Forms uses resources added to each individual user account using the Oracle Delegated Administration Services (DAS). 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 OID preferences. An administrator creates a resource once and all user accounts automatically inherit this resource to be used within Forms.

6.1.3 Support for Dynamic Directives With Forms SSO

Enforcing single sign-on in Forms is now done within the formsweb.cfg file. There is now a new SSO parameter, `mod_sso`, to indicate when a custom application requires SSO authentication.

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

For more information, see "ssoDynamicResourceCreate".

6.1.4 Support for Database Password Expiration for Forms Running with 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 (OID).

Now in Oracle Application Server Forms Services, if the database password has expired and the *Oracle9i*AS Forms Services application, running in 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 OID for this application. This feature ensures that 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 Forms, then the database connect string is not updated in OID.

6.2 Single Sign-on Components Used By Forms

The following software components in OracleAS are involved when running Forms applications in 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 in Release2. Oracle Application Server Forms Services and Oracle Application Server Reports Services use mod_osso to register as a partner application to the Oracle Application Server Single Sign-On Server
- Oracle Internet Directory (OID) - A LDAP v3 compliant directory server that stores user login information. An LDAP server is a special database that is optimized for read access.
- Forms Servlet - The Oracle Application Server Forms Services component that accepts the initial user request to start a Forms application. The Forms Servlet detects if an application requires single sign-on, directs the request to the Oracle Application Server Single Sign-On Server and accesses the Oracle Internet Directory to obtain the database connect information
- formsweb.cfg - The Forms configuration file that contains the parameters to enable a Forms application for single sign-on. The formsweb.cfg file is located in the forms90/server directory of an Oracle Application Server installation.

6.3 Enabling Single Sign-On for an Application

Oracle Forms applications are configured using a central configuration file, the formsweb.cfg file in the forms90/server directory. The formsweb.cfg file can be edited by using Enterprise Manager Application Server Control, which Oracle Corporation recommends.

Oracle Application Server 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 single sign-on definition overrides the same definition set in the **User Parameter** section.

6.3.1 ssoMode

The `ssoMode` parameter enables an Oracle Application Server Forms Services application for single sign-on. By default, Oracle Forms applications are not configured to run in single sign-on mode. The `ssoMode` parameter can be set in two places in the formsweb.cfg file. Setting `ssoMode` as a system parameter with a value of `true` allows all applications to run in 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 single sign-on only for this particular application:

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

To enable single sign-on for an application:

1. Start the Enterprise Manager Application Server Control.
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**.

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

To disable single sign-on for an application:

1. Start the Enterprise Manager Application Server Control.
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.

Optionally, when you set `ssoMode` to false, you can redirect users to an informational page by specifying a URL in the `ssoErrorUrl` parameter.

6.3.2 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 (OID) 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 (DAS). This Web page is not customizable as it is not owned by Oracle Forms.

Allowing dynamic resource creation simplifies OID 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 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 OID), these parameters allow administrators to control OID 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, see Chapter 4.3.3, "Managing Parameters".

6.3.3 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
…
```

6.3.4 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.

6.3.5 Accessing Single Sign-on Information From Forms

Optionally, if you need to work with single sign-on authentication information in a Forms application, the GET_APPLICATION_PROPERTY() built-in can be used to retrieve the following single sign-on login information: Oracle Application Server 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') ;
```

6.4 Availability of Information on Integrating Oracle Forms and Reports

Oracle Application Server Reports Services is installed with 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 single sign-on protected Oracle Forms application, the authenticated user's single sign-on identity is implicitly passed to the Reports Server with each call to RUN_REPORT_OBJECT() Built-in. The single sign-on identity is used to authenticate the user to the Reports Server for further authorization checking, if required.

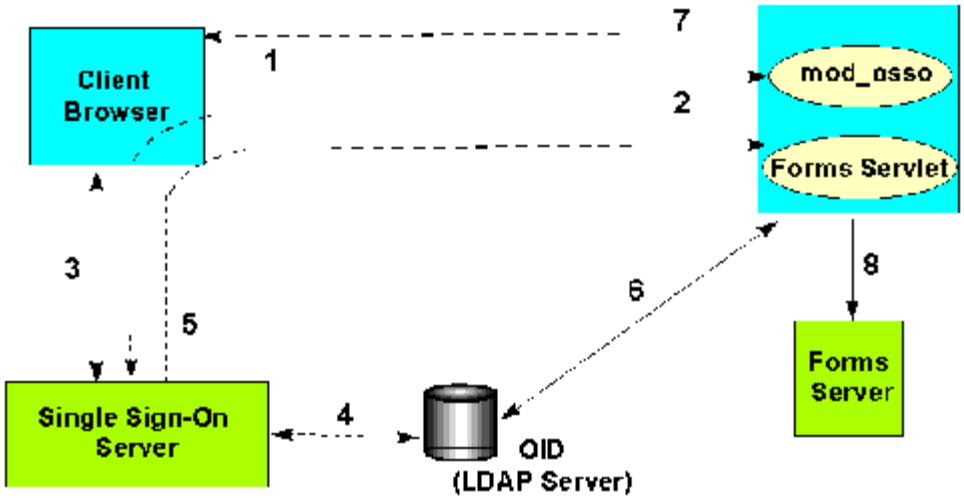
A Forms application running in non-single sign-on mode can run a report on a single sign-on secured Reports Server, but fails if the Reports Server requires authorization. Also users must provide their single sign-on credentials when retrieving the Reports output on the Web.

For more information about integrating Oracle Forms and Oracle Reports, see the white paper Integrating *Oracle 9iAS Report Services in Oracle 9iAS Forms Services* at Oracle Technology Network (http://otn.oracle.com/products/forms/).

6.5 Authentication Flow

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

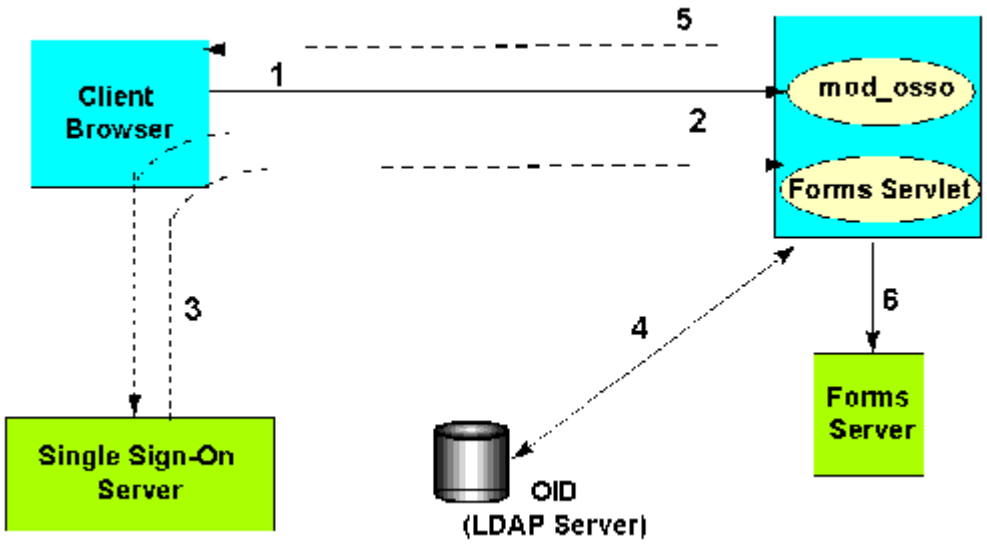
Figure 6–1 Authentication Flow for First Time Client Request



1. The user requests a Forms URL similar to `http(s)://<hostname>:<port>/forms90/f90servlet?config=<application>&...`
2. The Forms Servlet redirects the user to the SSO server.
3. The user provides user name and password through Login form.
4. The password is verified through OID (LDAP Server).
5. The user gets redirected to the URL with sso_userid information.
6. Forms Servlet gets the database credentials from OID.
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 Oracle Application Server 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

Tracing and Diagnostics

When you develop and deploy Oracle Forms applications, it is helpful to have information that allows you to optimize your applications. Tracing and diagnostic tools that are available with Oracle Forms allow you to analyze the performance and resource consumption of your Oracle Forms applications at runtime. Through Enterprise Manager Application Server Control, you can use trace output to diagnose performance and other problems with Oracle Forms applications.

The following tools are available to collect trace information for Oracle Forms:

- 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.
- Servlet Logging Tools: Enables site administrators to keep a record of all Oracle Forms sessions, monitor Oracle Forms-related network traffic, and debug site configuration problems.

7.1 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.

This section on Forms Trace contains the following information:

7.1.1 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 Application Server Control Forms page to define the events that you want to trace. This page manages all changes in the ftrace.cfg file for you.

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

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 Oracle Enterprise Manager Application Server Control to Forms configuration, trace, or environment files before proceeding to other pages.

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

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 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

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

Note: If you first switch off trace, and then switch it on again with new settings, then trace is enabled with the new trace group.

Note: In order to trace Forms Processes on Windows, the ProcessManager Service needs to have the check box "Allow service to interact with the desktop" checked. If 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.`

To configure Forms Trace:

1. Start the Enterprise Manager Application Server Control.
2. From the Enterprise Manager Application Server Control 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:

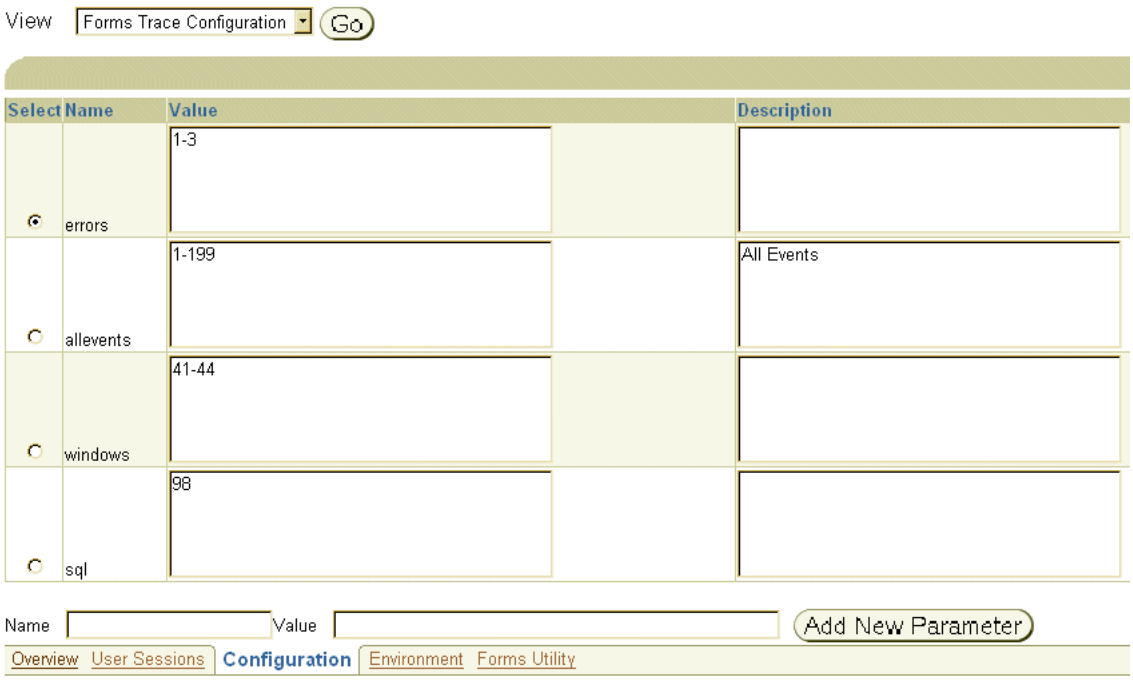
- Select the radio button next to it, and modify the values in the text areas. Click **Apply** to save your changes.

To save your changes:

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

The following is a sample ftrace.cfg configuration file where three event sets have been specified.

Figure 7–1 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"

7.1.1.1 Specifying URL Parameter Options

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

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

Table 7–1 Forms Trace Command Line Parameters

Parameter	Values	Description
Record	forms	Enables Forms Trace.

Table 7–1 (Cont.) Forms Trace Command Line Parameters

Parameter	Values	Description

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>/forms90/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.

7.1.2 Starting the Trace

You start a trace by specifying trace entries in the URL or from Enterprise Manager Application Server Control. 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/forms90/f90servlet?form=cxl&record=forms&tracegroup=0-199
http://cx-pc/forms90/f90servlet?form=cxl&record=forms&tracegroup=mysql
http://cx-pc/forms90/f90servlet?form=cxl&record=forms&tracegroup=0-199;log=run1.lo
g
```

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://otn.oracle.com/.

7.1.3 Viewing Forms Trace Output

Trace data is stored in a binary file with a *.trc extension. If you're not using Enterprise Manager Application Server Control, 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 Forms Enterprise Manager User Sessions screen in the Enterprise Manager Application Server Control.

To view trace data, use Enterprise Manager:

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

7.1.3.1 Running the Translate Utility

The Translate utility converts trace data to XML format.

To convert trace data to XML format:

- At the command line, enter:

```
java oracle.forms.diagnostics.Xlate datafile=a.trc xmlfile=myfile.xml
```

to create `myfile.xml`.

7.1.4 List of Traceable Events

The following table 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 7–2 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
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

Table 7–2 (Cont.) List of Traceable Events

Event Number	Definition	Type
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
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.

7.1.5 List of Event Details

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

- User Action Events
- Forms Services Events
- Detailed Events
- Three-Tier Events
- Miscellaneous

7.1.5.1 User Action Events

Table 7–3 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

7.1.5.2 Forms Services Events

Table 7–4 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

7.1.5.3 Detailed Events

Table 7–5 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

7.1.5.4 Three-Tier Events

Table 7–6 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

7.1.5.5 Miscellaneous

Table 7–7 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

7.1.6 Monitoring Forms Services Trace Metrics

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

1. Start the Enterprise Manager Application Server Control.
2. From the Enterprise Manager Application Server Control 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.

7.2 Servlet Logging Tools

The servlet logging tools available with Oracle Application Server Forms Services provides the following:

- A record 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)
- Information for debugging site configuration problems (debug-level logging)

This section on servlet logging tools contains the following information:

7.2.1 Enabling Logging

You enable logging by:

- Appending one of the strings in Table 7–8, " Supported logging capabilities" to the serverURL parameter in the URL that starts the form.
- Appending one of the strings in Table 7–8, " Supported logging capabilities" to the serverURL client parameter in the **Forms Configuration** page of Enterprise Manager Application Server Control.

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 Example Output for Each Level of Servlet Logging.

Table 7–8 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 machine 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.

7.2.1.1 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/forms90/f90servlet?serverURL=/forms90/l90servlet/perf
```

7.2.1.2 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=/forms90/f90servlet/session
```

7.2.1.3 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/forms90/f90servlet/debug?serverURL=/forms90/l90servlet/debug
```

7.2.2 Location of Log Files

The servlet log file is application.log. It is written to the application-deployments/forms90app 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/forms90app/OC4J_BI_
Forms_default_island_1/application.log
```

In Forms Developer, it is:

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

7.2.3 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

7.2.3.1 (none)

```
FormsServlet init():
configFileName:     d:\orant9i/forms90/server/formsweb.cfg
testMode:
  false
```

7.2.3.2 /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
```

7.2.3.3 /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
```

7.2.3.4 /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
```

7.2.3.5 /debug

Here is an example run by going to a URL like http://test-machine:8888/forms90/f90servlet/debug&config=ienative&serverURL=/forms90/l90servlet/debug):

```
============== FormsServlet ===============
GET request received, cmd=debug,
qstring=config=ienative&serverURL=/forms90/l90servlet/debug
No current servlet session
File baseie.htm not found, looking in d:\orant9i/forms90/server
The SSO_USERID is: null
============== FormsServlet ===============
GET request received, cmd=startsession, qstring=config=ienative&serverURL=
/forms90/l90servlet/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:\orant9i/forms90/server
envFile = d:\orant9i\forms90\server\default.env
serverURL: /forms90/l90servlet/debug
rewrittenURL: /forms90/l90servlet/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 ifweb90 server webfile=HTTP-0,0,1
Getting stdin, stdout and stderr of child process
Writing working directory to stdin: d:\orant9i\forms90
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.
**********************************************
```

8

Performance Tuning Considerations

This chapter describes the tuning considerations that arise when you deploy Oracle Forms applications to Oracle Application Server Forms Services. This chapter looks at the network and resources on the application server and includes the following sections:

- Built-in Optimization Features of Forms Services
- Tuning OracleAS Forms Services Applications

Tuning the connection between Oracle Application Server Forms Services and the *Oracle9i* Database Server is beyond the scope of this chapter.

8.1 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:

- Forms Services Web Runtime Pooling
- Minimizing Client Resource Requirements
- Minimizing Forms Services Resource Requirements
- Minimizing Network Usage
- Maximizing the Efficiency of Packets Sent Over the Network
- Rendering Application Displays Efficiently on the Client

8.1.1 Monitoring Forms Services

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

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

8.1.1.1 Monitoring Forms Services Instances

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

1. Start Enterprise Manager Application Server Control.

2. From the Enterprise Manager Application Server Control 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 of the Forms Services instance in milliseconds to connect to the Forms Listener Servlet.

   Additionally, you can jump to the following detail pages:

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

8.1.1.2 Monitoring Forms Events

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

8.1.1.3 Monitoring Metrics for User Sessions

1. Start the Enterprise Manager Application Server Control.
2. From the Enterprise Manager Application Server Control 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 the Forms runtime process for 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 machine 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 Configuration Section Parameters page for the configuration section used by particular forms session .

8.1.1.4 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.

8.1.1.5 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**.

8.1.2 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.

8.1.2.1 Configuring Prestart Parameters

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

Table 8–1 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 8–1 (Cont.) Forms Runtime Pooling Parameters

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

Each configuration section can specify values for these parameters. 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.

8.1.2.2 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. 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.

8.1.3 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.

8.1.3.1 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.

8.1.4 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.

8.1.5 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.

8.1.6 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.

8.1.7 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.

8.1.8 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.

8.2 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.

Issues discussed are:

- Location of the Oracle Application Server Forms Services with Respect to the Data Server
- Minimizing the Application Startup Time
- Reducing the Required Network Bandwidth
- Other Techniques to Improve Performance

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

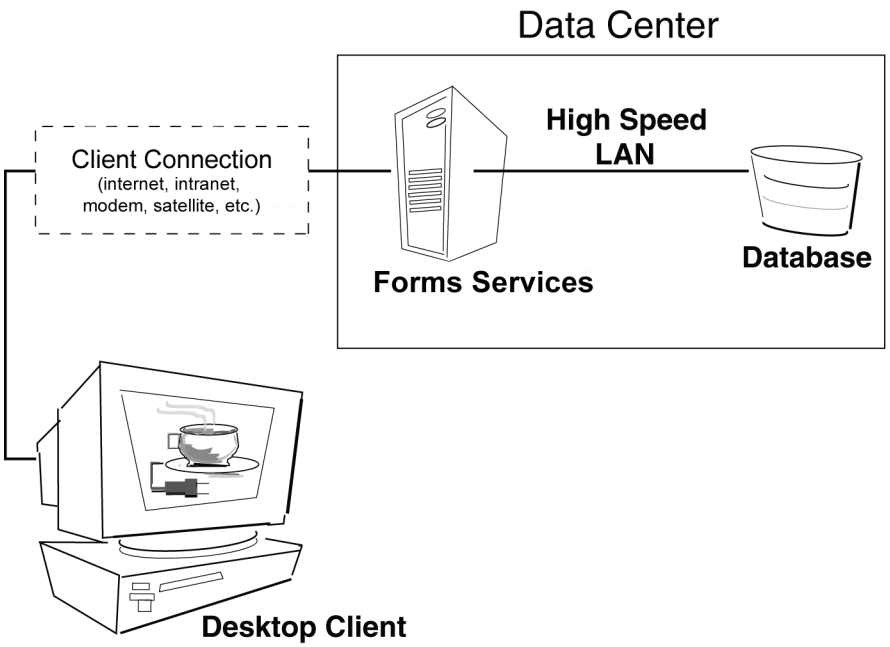
The Forms Java client is only responsible to display the GUI objects. All of the Oracle Forms logic runs in Oracle Application Server Forms Services, on the middle tier. This includes inserting or updating the data to the database, querying data from the database, executing stored procedures on the database, 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

Oracle Application Server Forms Services. This allows Oracle Forms applications to run across slower networks, such as with modems or satellites.

The configuration in Figure 8–1, displays how Oracle Application Server Forms Services and the database server are co-located in a data center.

Figure 8–1 Co-Locating the OracleAS Forms Services and Database Server



8.2.2 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

8.2.2.1 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.

8.2.2.1.1 Oracle JInitiator

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

- f90all.jar - includes all required classes
- f90all_jinit.jar - same as f90all.jar but is optimized for use with Oracle JInitiator (this is the default)
- f90main.jar - contains fewer classes than f90all.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=f90all_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 f90main.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 f90all.jar and f90all_jinit.jar, so they are only used if you use f90main.jar.

- `f90oracle_laf.jar` – classes for the Oracle Look-And-Feel
- `f90generic_laf.jar` – classes for the generic (standard) Look-And-Feel
- `f90resources.jar` – resource classes for languages other than US English.

  The English resource classes are contained in `f90all.jar`, `f90all_jinit.jar`, and `f90main.jar`. `f90resources.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 A, "JInitiator".

8.2.2.1.2 IE Native JVM

Since IE does not support Jar signing, you will need to use a CAB file. The following CAB file is provided for use with IE:

- `f90all.cab` - includes all required classes

While `f90all.cab` is the only file provided for use with IE, it is significantly smaller than `f90all.jar`.

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

```
[MyApp]
archive_ie=f90all.cab
```

8.2.2.1.3 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:

- `f90all.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=f90all.jar
```

8.2.2.2 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.

8.2.3 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)
VISIBLE = NO
```

Pay attention to TAB canvases that consist of several sheets where only one will ever be displayed. For responsive switching between tabs, all items for all sheets on the canvas are loaded, including those that are hidden behind the initial tab. Consequently, the time taken to load and initialize a TAB canvas is related to all objects on the canvas and not just to those initially visible.

Tip: When using Tab canvases, use stacked canvases and display the right canvas in the when-tab-page-changed trigger. Remember to set the properties `RAISE ON ENTRY = YES` and `VISIBLE = NO` for all the canvases not displayed in the first screen.

- **Disable MENU_BUFFERING.** By default, MENU_BUFFERING is set to True. This means that changes to a menu are buffered for a future "synchronize" event when the altered menu is re-transmitted in full. (Most applications make either many simultaneous changes to a menu or none at all. Therefore, sending the entire menu at once is the most efficient method of updating the menu on the client.) However, a given application may make only minimal changes to a menu. In this case, it may be more efficient to send each change as it happens. You can achieve this using the statement:

```
Set_Application_Property (MENU_BUFFERING, 'false');
```

Menu buffering applies only to the menu properties of LABEL, ICON, VISIBLE, and CHECKED. An ENABLE/DISABLE event is always sent and does not entail the retransmission of an entire menu.

8.2.4 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' machines.

8.3 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 (OHS) instance and OC4J instance on Host B running Forms9i application D
3. OHS instance and OC4J instance on Host C running Oracle Forms application D

Note that there could be more OHS/OC4J instances, but only two instance pairs will be described here for purposes of simplification. The OHS/OC4J instances are not clustered because Oracle Forms applications cannot take advantage of iAS 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 OHS instances running on Hosts B and C. When configuring Origin Server info for Hosts B and C, be sure to configure a ping URL that will detect whether Forms application D is running, for example, `/forms90/f90servlet?ifcmd=status`.

To Configure Session Binding in Web Cache:

1. Log on to the Web Cache Manager.
2. In the navigator pane, select **Origin Servers, Sites, and Load Balancing | Session Binding**.
3. In the **Session Binding** screen, select **Default Session Binding**, then select **Edit Selected**.
4. The **Edit Session Binding** dialog box appears.
5. From the **Please select a session:** pull-down list, select **Monitoring | Health Monitor**.
6. Configure an **Inactivity Timeout** that is appropriate for Oracle Forms application D.
7. Click **Submit**.
8. 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 OHS/OC4J that handled the requests. For example, assume this is Host B and shut down the OHS/OC4J on that host. Now only the OHS/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 OHS/OC4J running on Host C. Ensure that the application works as expected.
5. Restart the OHS/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 *OracleAS Web Cache Administration and Deployment Guide*.

A

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 (f90all.jar and f90all_jinit.jar). f90all.jar is a standard Jar file, and f90all_jinit.jar is a Jar file with extra compression that can only be used with Oracle JInitiator.

A.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).

A.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.

A.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 machine 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.

A.4 Supported Configurations

Oracle JInitiator supports the following configurations:

A.4.1 Windows 98, NT, 2000, XP:

- Navigator 4.7.x
- Navigator 4.7.x
- Internet Explorer 5.x
- Internet Explorer 6.0

A.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)

A.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.

A.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 Corporation. 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.

A.8 Setting up the Oracle JInitator Plug-in

To set up the Oracle JInitiator plug-in:

- Add Oracle JInitiator HTML markup to your base HTML file.
- Install Oracle JInitiator on your server (for server-based testing purposes only).
- Customize the Oracle JInitiator download file.
- Make Oracle JInitiator available for download.

A.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 B.3, "base.htm, basejini.htm, basejpi.htm, and baseie.htm Files".

A.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.

A.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.

1. 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.

A.9 Modifying the Oracle JInitiator plug-in

To modify the Oracle JInitiator plug-in:

- Modify the cache size for Oracle JInitiator.
- Modify the heap size for Oracle JInitiator.
- Check and modify the proxy server setting for **Oracle JInitiator**.
- View Oracle JInitiator output.

A.9.1 Modifying the cache size for 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.

The default cache size for Oracle JInitiator is 20000000. This is set for you when you install Oracle JInitiator.

A.9.2 Modifying the heap size for 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.

The default maximum heap size for Oracle JInitiator is 64MB. This has been set for you when you install Oracle JInitiator.

A.9.3 Check and modify 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 checked. Then, enter the host name for the proxy server in the **Proxy Address** field.

A.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. Check the **Show Java Console** check box to enable debug output.

A.10 Modifying the baseHTML file

When you run an Oracle Forms application with the help of JInitiator, JIinitiator 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 B.3, "base.htm, basejini.htm, basejpi.htm, and baseie.htm Files". Adding these tags to your baseHTML file will enable your applications to run within both Netscape and Microsoft browsers.

B

Sample Configuration Files

During the installation, the following configuration files were installed onto your system:

- Default formsweb.cfg File
- Platform Specific default.env Files
- base.htm, basejini.htm, basejpi.htm, and baseie.htm Files
- web.xml
- forms90.conf
- Registry.dat

B.1 Default formsweb.cfg File

The default formsweb.cfg file contains the following:

```
# formsweb.cfg defines parameter values used by the FormsServlet (f90servlet)
# 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 overriden 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/servlet/f90servlet?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: base HTML file for use with Microsoft Internet Explorer
# (when using the native JVM)
baseHTMLie=baseie.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>/forms90 if unset.
workingDirectory=
# System parameter: file setting environment variables for the Forms runtime
# processes
envFile=default.env
# System parameter: JVM option for Microsoft Internet Explorer.
# This parameter specifies how to execute the Forms applet under
# Microsoft Internet Explorer 5.x or above.  Put IE=native if you want
# the Forms applet to run in the browser's native JVM.
IE=JInitiator
# 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=/forms90/l90servlet
# Forms applet parameter
codebase=/forms90/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=f90all_jinit.jar
# Forms applet archive setting for Microsoft Internet Explorer native JVM
archive_ie=f90all.cab
# Forms applet archive setting for other clients (Sun Java Plugin, Appletviewer,
# etc)
archive=f90all.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=/forms90/jinitiator/us/jinit_download.htm
# Parameter related to the version of JInitiator
jinit_classid=clsid:CAFECAFE-0013-0001-0017-ABCDEFABCDEF
# Parameter related to the version of JInitiator
jinit_exename=jinit.exe#Version=1,3,1,17
# Parameter related to the version of JInitiator
jinit_mimetype=application/x-jinit-applet;version=1.3.1.17
# 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/plugin/1.3/plugin-install.html
# Parameter related to the version of the Java Plugin
```

```
jpi_classid=clsid:8AD9C840-044E-11D1-B3E9-00805F499D93
# Parameter related to the version of the Java Plugin
jpi_
codebase=http://java.sun.com/products/plugin/1.3/jinstall-13-win32.cab#Version=1,3
,0,0
# Parameter related to the version of the Java Plugin
jpi_mimetype=application/x-java-applet;version=1.3

# 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=formsApp_adtqa_ui7.us.oracle.com_6A5E0A34DCD44048B706A0ECE46EC3A6
# Single Sign-On OID configuration parameter
oracle_home=D:\AS10g_M30_bif
# Single Sign-On OID configuration parameter
formsid_group_dn=cn=Logical Application Group, orclApplicationCommonName=formsApp_
adtqa_ui7.us.oracle.com_6A5E0A34DCD44048B706A0ECE46EC3A6, cn=forms, cn=Products,
cn=OracleContext
# 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
# 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/servlet/f90servlet?config=sepwin".
[sepwin]
separateFrame=True
lookandfeel=Generic
# Example Named Configuration Section
# Example 2: configuration affecting users of MicroSoft Internet Explorer 5.x.
# Forms applet will run under the browser's native JVM rather than using Oracle
JInitiator.
[ienative]
IE=native
```

```
# Example Named Configuration Section
# Example 3: configuration forcing use of the Java Plugin in all cases (even if
# the client browser is on Windows)
[jpi]
baseHTMLJInitiator=basejpi.htm
baseHTMLie=basejpi.htm
# Example Named Configuration Section
# Example 4: configuration running the Forms ListenerServlet in debug mode
# (debug messages will be written to the servlet engine's log file).
[debug]
serverURL=/forms90/l90servlet/debug
```

B.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 UNIX

B.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=<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. /private/dir1;/private/dir2)
  #
  # FORMS90_PATH=<ORACLE_HOME>/forms90

  #
  # 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=<ORACLE_HOME>\bin;<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.

    #FORMS90_TRACE_PATH=<ORACLE_HOME>\forms90\server


  #
  # System settings
  # ---------------
  # You should not normally need to modify these settings
  #
  FORMS90=<ORACLE_HOME>\forms90

  #
  # Java class path
  #   This is required for the Forms debugger
  #   You can append your own Java code here)
  # f90srv.jar, repository.jar and ldapjclnt9.jar are required for
  # the password expiry feature to work(#2213140).
  #
  CLASSPATH=<ORACLE_HOME>\j2ee\OC4J_BI_
Forms\applications\forms90app\forms90web\WEB-INF\lib\f90srv.jar;<ORACLE_
HOME>\jlib\repository.jar;<ORACLE_HOME>\jlib\ldapjclnt9.jar;<ORACLE_
HOME>\jlib\debugger.jar;<ORACLE_HOME>\jlib\ewt3.jar;<ORACLE_
HOME>\jlib\share.jar;<ORACLE_HOME>\jlib\utj90.jar;<ORACLE_
HOME>\jlib\zrclient.jar;<ORACLE_HOME>\reports\jlib\rwrun.jar
```

B.2.2 Default default.env File for UNIX

```
# 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=<ORACLE_HOME>

  #
  # Search path for Forms applications (.fmx files, PL/SQL libraries)
  #
  FORMS90_PATH=<ORACLE_HOME>/forms90

  # Java class path
  #   This is required for the Forms debugger
  #   You can append your own Java code here)
  # f90srv.jar, repository.jar and ldapjclnt9.jar are required for
  # the password expiry feature to work(#2213140).
  #
  CLASSPATH=<ORACLE_HOME>\j2ee\OC4J_BI_
Forms\applications\forms90app\forms90web\WEB-INF\lib\f90srv.jar;<ORACLE_
HOME>\jlib\repository.jar;<ORACLE_HOME>\jlib\ldapjclnt9.jar;<ORACLE_
HOME>\jlib\debugger.jar;<ORACLE_HOME>\jlib\ewt3.jar;<ORACLE_
HOME>\jlib\share.jar;<ORACLE_HOME>\jlib\utj90.jar;<ORACLE_
HOME>\jlib\zrclient.jar;<ORACLE_HOME>\reports\jlib\rwrun.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=<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.
REPORTS_CLASSPATH=<ORACLE_HOME>/jlib/zrclient.jar:<ORACLE_
HOME>/reports/jlib/rwrun.jar

  #
  #
  # 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.

    #FORMS90_TRACE_PATH=<ORACLE_HOME>/forms90/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
    #     (/cdm/solaris/o_jdk/1_2_2_0_0/jre/lib/sparc)
    #   - Forms needs two paths to the jre, for libjvm.so and libhpi.so
    #   - In ojdk 1.3.1 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=<ORACLE_HOME>/lib:%O_JDK_HOME%/jre/lib/sparc:%O_JDK_
HOME%/jre/lib/sparc/native_threads:%LD_LIBRARY_PATH%
```

B.3 base.htm, basejini.htm, basejpi.htm, and baseie.htm Files

For a brief description and the locations of base.htm, basejini.htm, basejpi.htm, and baseie.htm, see Chapter 3.1.1.3, "base.htm, basejini.htm, basejpi.htm, and baseie.htm".

Four baseHTML files are created for your system by the Oracle Universal Installer during OracleAS installation and configuration. **In most cases, you will not need to modify these files.** If you do need to modify these files, you should create your own versions and reference them from the formsweb.cfg file. The default files may be overridden by a patch installation.

When a user first starts an Oracle Forms application (by clicking a link to the application's URL), a baseHTML file is read by Forms Servlet.

Any variables (%*variablename*%) in the baseHTML file are replaced with the appropriate parameter values specified in the formsweb.cfg file described in Section 4.3.1, "Configuring Parameters with Application Server Control", 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 Section 4.3.1, "Configuring Parameters with Application Server Control" file.

The following baseHTML starter files are available in the <ORACLE_HOME>/forms90/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.
- **baseie.htm**: This is a baseHTML file containing the Internet Explorer 5 tags required to use native JVM in Internet Explorer 5. See Default baseie.htm File for an example.

To create a new baseHTML file:

1. Place the new baseHTML file in any directory. Update the basejini.htm, baseie.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, baseie.htm, basejpi.htm, or base.htm starter file, which is located in the <ORACLE_HOME>/forms90/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 Section 3.1.1.2, "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.

B.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.

B.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.

B.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%">

</APPLET>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

B.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="/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%">
<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%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

B.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%">
<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%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

B.3.5 Default baseie.htm File

```
<HTML>
<!-- FILE: baseie.htm (Oracle Forms)                                    -->
<!--                                                                    -->
<!-- This is the default base HTML file for running a form on the       -->
<!-- web with Internet Explorer version 5.0 or above, using that       -->
<!-- browser's native JVM.                                              -->
<!--                                                                    -->
<!-- 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           -->
<!-- baseHTMLie 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"
        WIDTH="%Width%"
        HEIGHT="%Height%">

<PARAM NAME="cabbase" VALUE="%archive_ie%">
<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%">

</APPLET>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

B.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 (OracleDS) 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 f90servlet 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>/forms90/f90servlet/admin).

- Use a Forms Servlet configuration file other than the standard one (which is <ORACLE_HOME>/forms90/server/formsweb.cfg).

  This can be done by uncommenting and changing the f90servlet'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).

B.4.1 Default web.xml File

```
- <web-app>
  <display-name>Forms Services</display-name>
  <description>OracleAS: Forms Services</description>
- <welcome-file-list>
  <welcome-file>l90servlet</welcome-file>
  </welcome-file-list>
- <!--  Forms page generator servlet
  -->
- <servlet>
```

```
  <servlet-name>f90servlet</servlet-name>
  <servlet-class>oracle.forms.servlet.FormsServlet</servlet-class>
- <!--
 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>/forms90/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 f90servlet/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>l90servlet</servlet-name>
  <servlet-class>oracle.forms.servlet.ListenerServlet</servlet-class>
  </servlet>
- <!--
 Forms servlet mappings. Allow these paths to the servlets:
           /forms90/f90servlet or /forms90/f90servlet/*: FormsServlet
           /forms90/l90servlet or /forms90/l90servlet/*: ListenerServlet


  -->
- <servlet-mapping>
  <servlet-name>f90servlet</servlet-name>
  <url-pattern>/f90servlet*</url-pattern>
  </servlet-mapping>
- <servlet-mapping>
  <servlet-name>l90servlet</servlet-name>
  <url-pattern>/l90servlet*</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 />
  </context-param>
  </web-app>
```

B.5 forms90.conf

For a description and the location of forms90.conf, see Section 3.1.3.1, "forms90.conf".

The following table describes the virtual paths and servlet mappings:

Table B–1 forms90.conf Virtual Paths and Servlet Mappings

URL Path	Type	Maps to	Purpose
/forms90/java	Alias	<ORACLE_HOME>/forms90/java	codebase for Forms applet. Used to download the applet code to the user's web browser.
/forms90/html	Alias	<ORACLE_HOME>/tools/web90/html	Access runform.htm (used to run any form for testing)
/forms90/jinitiator	Alias	<ORACLE_HOME>/jinit	Oracle JInitiator download
/forms90/f90servlet	Servlet mount point	Forms Servlet	Generate HTML page to run a form
/forms90/l90servlet	Servlet mount point	Forms Listener Servlet	Handles message traffic from the Forms applet

B.5.1 Default forms90.conf

```
# Name
#   forms90.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 "forms90" servlet zone with properties file forms90.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 ^/forms90/java/(..*) "<ORACLE_HOME>/forms90/java/$1"

  # Virtual path for JInitiator downloadable executable and download page
AliasMatch ^/jinitiator/(..*) "<ORACLE_HOME>/jinit/$1"

  # Virtual path for runform.htm (used to run a form for testing purposes)
AliasMatch ^/forms90/html/(..*) "<ORACLE_HOME>/tools/web90/html/$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 /forms90/f90servlet
    # and /forms90/l90servlet respectively.
    # We map f90servlet to servlet.if90, and l90servlet to servlet.ifl90.
    # The apJServAction directives (below) will then remap those.
    AliasMatch ^/forms90/f90servlet(.*) "/servlet.if90"
    AliasMatch ^/forms90/l90servlet(.*) "/servlet.ifl90"
    ApJServMount /forms90/servlet /forms90
    #
    # Let the servlets be called by file extension (e.g /servlet.if90)
    #
    ApJServAction .if90 /forms90/servlet/f90servlet
    ApJServAction .ifl90 /forms90/servlet/l90servlet
    # Prevent access to the Forms Servlets by paths other than
    # /forms90/f90servlet and /forms90/l90servlet.
    # 1. Prevent access via the .if90 and .ifl90 file extensions:
    <LocationMatch ^.*\.if.*90>
      order deny,allow
      deny from all
    </LocationMatch>
    # 2. Stop access by class (by paths like
    #    /forms90/servlet/oracle.forms.servlet.FormsServlet)
    <LocationMatch ^/forms90/servlet/oracle\.forms.*>
      order deny,allow
      deny from all
    </LocationMatch>
  </IfModule>
</IfModule>

# Config. for OC4J
<IfModule mod_oc4j.c>
    Oc4jMount /forms90              OC4J_BI_Forms
    Oc4jMount /forms90/f90servlet   OC4J_BI_Forms
    Oc4jMount /forms90/f90servlet/* OC4J_BI_Forms
    Oc4jMount /forms90/l90servlet   OC4J_BI_Forms
    Oc4jMount /forms90/l90servlet/* OC4J_BI_Forms
</IfModule>
```

B.6 Registry.dat

For a description and the location of Registry.dat, see Chapter 3.1.4.1, "Registry.dat".

The main reason you would want to edit this file is to change the icon settings (see Section 4.9.1, "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,D
ialog,Dialog,Serif,Serif,Dialog,SansSerif
```

Some fonts on NT 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 Section B.6.1, "Default Registry.dat"

B.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,D
ialog,Dialog,Serif,Serif,Dialog,SansSerif

#
# The Application Level icon files are relative to the DOCUMENTBASE
#   example: icons/
# or an absolute URL.
#   example: http://www.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
```

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W