Oracle9*i*AS Forms Services

Deployment Guide

Release 9.0.2

January 2002

Part No. A92175-01

ORACLE

Oracle9iAS Forms Services Deployment Guide, Release 9.0.2 for Windows and Solaris

Part No. A92175-01

Contributing Authors: Orlando Cordero, Tom Pfaeffle, Dean Ho, Duncan Mills, Frank Nimphius, Cathy Godwin, Bryan Roberts

Contents

3 Configuring Oracle9iAS Forms Services

4 Using Oracle9iAS Forms Services with the HTTP Listener and OC4J

5 Using Oracle9iAS Forms Services with SSO and OID

6 Enterprise Manager and Oracle9i Forms

7 Tracing and Diagnostics

8 Performance Tuning Considerations

A JInitiator

Index

List of Figures

List of Tables

Send Us Your Comments

Oracle9iAS Forms Services Deployment Guide, Release 9.0.2

Part No. A92175-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

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

- Electronic mail: oddoc_us@oracle.com
- Postal service:
  Oracle Corporation
  Oracle9*i* Forms Developer and Oracle9*i*AS Forms Services Documentation
  200 Oracle Parkway, 2op981
  Redwood Shores, CA 94065
  USA

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

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 Oracle9*i* Forms applications to the Web with the Oracle9*i* Application Server.

Structure

This manual contains the following chapters and appendices:

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 **Basics of Deploying Oracle9i Forms Applications**

Introduces you to the basic files you need to configure Oracle9*i*AS Forms Services and describes the steps for deploying applications.

Chapter 3 **Configuring Oracle9iAS Forms Services**

Describes advanced topics in configuring Oracle9*i*AS Forms Services and provides guidelines and tips for designing Oracle9*i* Forms applications for Web deployment.

Chapter 4 **Using Oracle9iAS Forms Services with the HTTP Listener and OC4J**

Describes how Oracle9*i*AS Forms Services works using Oracle HTTP Listener and OC4J.

Chapter 5 **Using Oracle9iAS Forms Services with SSO and OID**

Describes using Oracle9*i*AS Forms Services with single sign-on and Oracle Internet Directory.

Chapter 6 **Enterprise Manager and Oracle9i Forms**

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

Chapter 7 **Tracing and Diagnostics**

Describes tracing and diagnostic tools that are available with Forms allow you to analyze the performance and resource consumption of your Oracle9*i* Forms applications at runtime.

Chapter 8 **Performance Tuning Considerations**

Describes the tuning considerations when you deploy an application using Oracle9*i*AS Forms Services.

Appendix A **JInitiator**

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

Related Documents

For more information, see the following manuals:

- *Oracle9i Forms Developer and Forms Services: Release Notes for Windows (Part No. A92176-01)*
- *Oracle9i Forms Developer and Forms Services: Release Notes for Solaris (Part No. A92187-01)*
- *Oracle9i Forms Developer and Forms Services: Migrating Forms Applications from Forms6i (Part No. A92183-01)*
- Forms Developer Online Help, available from the Help menu in Forms Developer.

1

Introduction

Introduction

This guide is intended to provide information about deploying applications with Oracle9*i*AS 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:

- The Oracle Internet Platform
- Oracle 9iAS Forms Services
- Oracle9iAS Forms Services Architecture
- Oracle9iAS Forms Services Components
- Forms Listener Servlet

The Oracle Internet Platform

With Oracle9*i* database to manage data, Oracle9*i* Developer Suite (Oracle9*i*DS) to build applications, and Oracle9*i* Application Server (Oracle9*i*AS) 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:

- Oracle9i Application Server (Oracle9iAS)
- Oracle9i Developer Suite (Oracle9iDS)
- Oracle9i Database

Oracle9*i* Application Server (Oracle9*i*AS)

Oracle9*i*AS is a scalable, secure, middle-tier aplication server. It enables you to deliver Web content, host Web applications, and connect to to back-office applications. Forms Services are an integral part of the Oracle9*i* Application Server bundle, which provides the technology to fully realize the benefits of Internet computing.

Oracle9*i* Developer Suite (Oracle9*i*DS)

Oracle9*i*DS combines the power of Oracle Application Development tools, Oracle Business Intelligence tools, the award-winning XML Developer's Kit (XDK) and the Portlet Development Kit (PDK) in one product.

Oracle9*i*DS is based on Internet standards including J2EE, XML, SOAP, UDDI, and UML, and provides a highly productive environment to build applications for Oracle9*i*AS and the Oracle9*i* Database.

Oracle9*i* Database

Oracle9*i* Database 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.

Oracle9*i* Application Server is the best application server for the Oracle9*i* Database and applications built with Oracle development tools. By leveraging a common technology stack, Oracle9*i* Application Server can transparently scale an Oracle database by caching data and application logic on the middle tier.

Oracle 9*i*AS Forms Services

As part of Oracle9*i*AS, Oracle9*i*AS Forms Services is a new generation of tools that enable you to deploy new and existing Oracle9*i* Forms applications on the World Wide Web.

Oracle9*i*AS Forms Services is a comprehensive application framework optimized to deploy Oracle9*i* 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.

What's New in Oracle9*i*AS Forms Services?

Much of the functionality that was handled by the Web Server in Web6*i* has been assumed by components that are delivered with Oracle9*i*AS. For example, load balancing, security, scalability, HTTP/S communication handling, and deployment of Java servlets are all performed by various components delivered with Oracle9*i*AS, such as the Oracle HTTP Server and Oracle9*i*AS Containers for J2EE (OC4J).

The Oracle9*i*AS Forms Services component of Oracle9*i*AS handles all processing that is specific to Oracle9*i* Forms applications, such as running the business logic defined in the Oracle9*i* Forms application and providing the connection to the Oracle database. A Java applet provides the client user interface.

New features for Oracle9*i*AS Forms Services include:

- Support for deployment on the Web using the Forms Listener Servlet.
- Integration with OC4J (see Chapter 4, "Using Oracle9iAS Forms Services with the HTTP Listener and OC4J")
- Single Sign-On (see Chapter 5, "Using Oracle9iAS Forms Services with SSO and OID")
- Integration with OID (see Chapter 5, "Using Oracle9iAS Forms Services with SSO and OID")
- Integration with Enterprise Manager for easier administration and manageability (see Chapter 6, "Enterprise Manager and Oracle9i Forms")
- Tracing and logging improvements (see Chapter 7, "Tracing and Diagnostics")

Oracle9*i*AS Forms Services Architecture

Oracle9*i*AS Forms Services use a three-tier architecture to deploy database applications. Figure 1–1 shows the three tiers that make up the Oracle9*i*AS 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

.

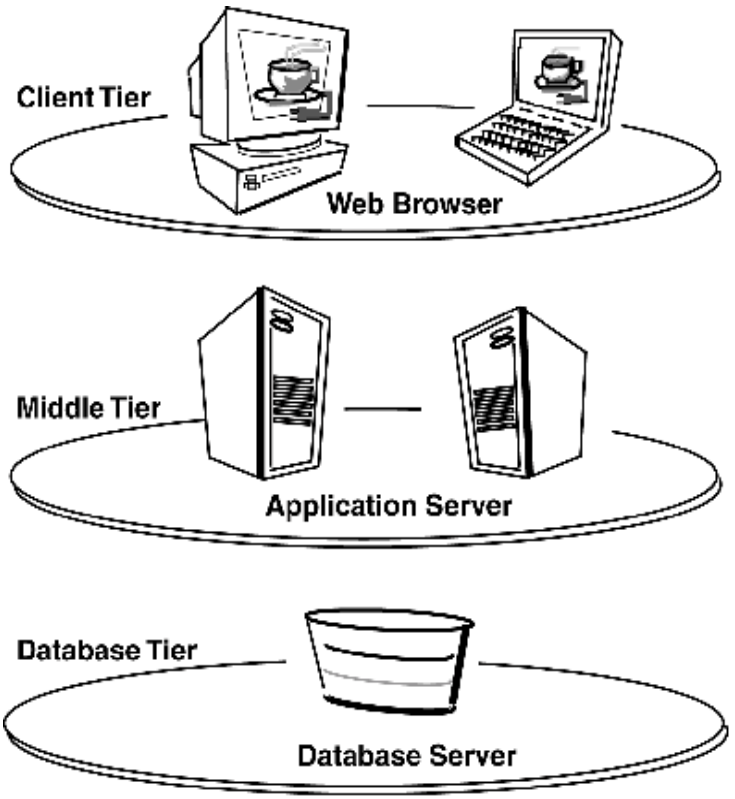


Figure 1–1 Oracle 9iAS Forms Services architecture

Oracle9*i*AS Forms Services Components

Oracle9*i*AS Forms Services is a middle-tier application framework for deploying complex, transactional forms applications to the Internet. Developers can build new applications with Oracle9*i* Forms Developer and deploy them to the Internet with Oracle9*i*AS 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.

Oracle9*i*AS 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

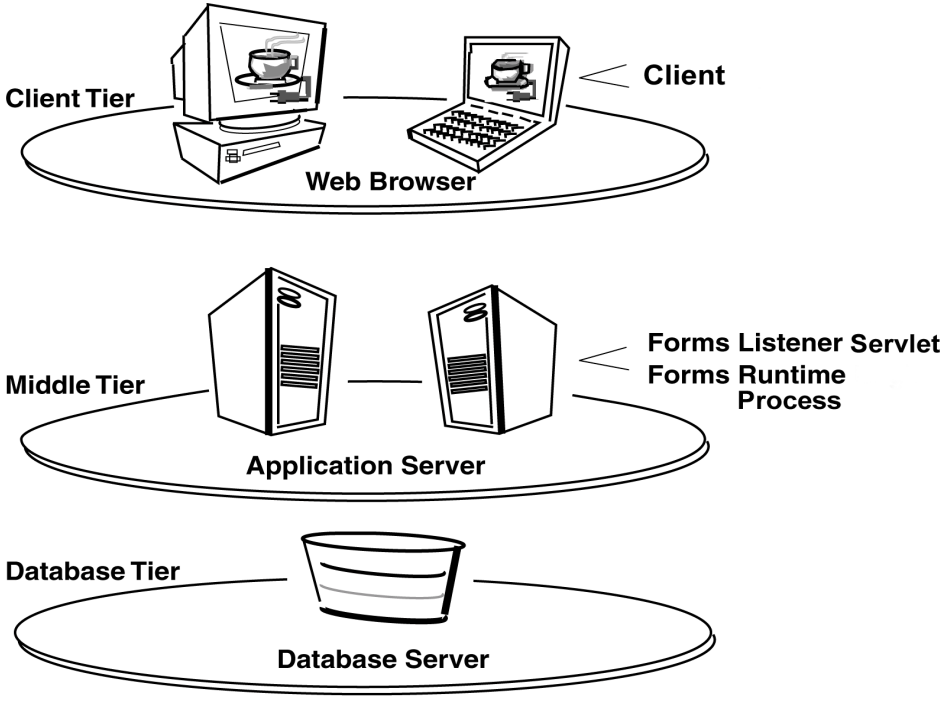


Figure 1–2 Three-tier configuration for running a form

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.

For more information, see Forms Listener Servlet.

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.

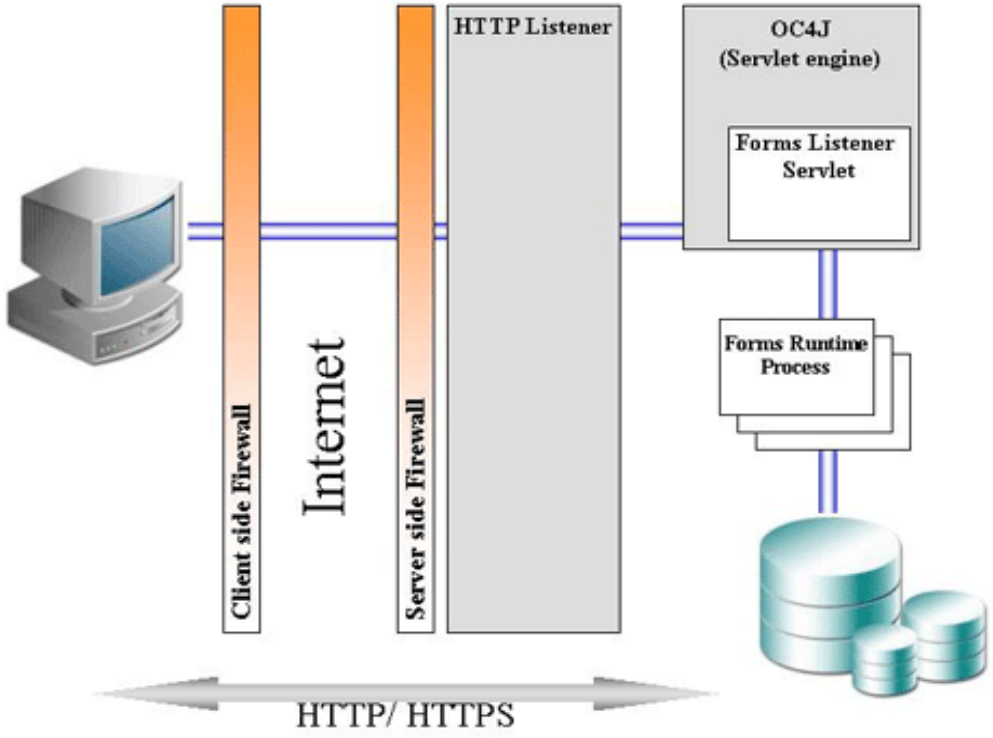
Forms Listener Servlet

Oracle9*i*AS 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 Oracle9*i* 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 Oracle9*i* Forms.

Note: You do not need to configure the Forms Listener Servlet as it is already set up for you in the Oracle9*i*AS 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

Basics of Deploying Oracle9*i* Forms Applications

Introduction

This chapter describes the basic files you need to configure Oracle9*i*AS Forms Services, provides an overview of how Oracle9*i*AS Forms Services runs in Oracle9*i*AS, and describes the steps you need to follow to deploy Oracle9*i* 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:

- Configuration Files
- Application Deployment
- Oracle9iAS Forms Services in Action
- Client Browser Support

Configuration Files

This section introduces the basic files you need to configure Oracle9*i* Forms applications. For more advanced configuration topics, see Chapter 3, "Configuring Oracle9iAS Forms Services".

This section contains the following sub-sections:

- Oracle 9i Forms Configuration Files
- Oracle9iAS Containers for J2EE (OC4J) Configuration Files
- Oracle HTTP Listener Configuration Files

- Standard Fonts and Icons File
- Environment Variables

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

Oracle 9*i* Forms Configuration Files

Oracle 9*i* Forms configuration files allow you to specify parameters for your forms. This section contains the following sub-sections:

- formsweb.cfg
- base.htm, basejini.htm, basejpi.htm, and baseie.htm
- ftrace.cfg

formsweb.cfg

Location: *forms90/server*.

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

- Values for Forms applet and 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).

For more information about formsweb.cfg, see Chapter 3, formsweb.cfg.

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 Oracle9*i* 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 more information about baseHTML files, see Chapter 3, base.htm, basejini.htm, basejpi.htm, and baseie.htm.

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 9*i* Forms. Forms Trace allows you to trace the execution path through a form (for example, steps the user took while using the form).

For more information about ftrace.cfg, see Chapter 7, ftrace.cfg.

Oracle9*i*AS Containers for J2EE (OC4J) Configuration Files

By default Oracle9*i* Forms 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 Oracle9*i*AS installation process (if you choose to configure Oracle 9*i* Forms). During deployment, the EAR file is unpacked into the applications directory of the OC4J instance.

This section describes:

- web.xml
- Directory structure for Oracle9i Forms OC4J files

web.xml

Location: *j2ee/ProductGroup2/applications/forms90app/forms90web/WEB-INF/web.xml*.

Once Oracle9*i* Forms has been installed and configured, the web.xml file is located in the directory
j2ee / ProductGroup2 / 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 3, web.xml.

Directory structure for Oracle9*i* Forms OC4J files

During Oracle9*i*AS installation and configuration, the Forms EAR file (forms90app.ear) is deployed to the "ProductGroup2" OC4J instance. This results in the following directory structure.

Names with a + sign are directories:

%ORACLE_HOME%/j2ee/ProductGroup2/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)

Oracle HTTP Listener Configuration Files

This section describes the file used to configure Oracle HTTP Listener for Oracle9*i* Forms.

forms90.conf

Location: *forms90/server*.

This is the Oracle HTTP listener configuration file for Oracle9*i* Forms. 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 3, forms90.conf.

Standard Fonts and Icons File

This section describes the file used to configure font and icon settings for Oracle9*i* Forms.

registry.dat

Location: *forms90/java/oracle/forms/registry*

This file allows you to change the default font, font mappings, and icons that Oracle9*i*AS Forms Services uses.

For more information about registry.dat, see Chapter 3, registry.dat.

Environment Variables

This section describes the default environment variable containing environment settings for the Forms runtime process.

For a full list of customizable environment variables, see Chapter 3, Customizing Environment Variables and Registry Settings.

default.env

Location: *forms90/server.*

This file contains environment settings for Forms runtime and can be found in the same directory as the formsweb.cfg file. On UNIX, default.env should include the PATH and LD_LIBRARY_PATH.

For more information about default.env, see Chapter 3, default.env.

Application Deployment

Once you have created your application in Oracle9*i* Forms Developer , you are ready for application Web deployment. Oracle9*i* Forms accesses an application in Oracle9*i*AS through a specified URL. The URL then accesses the HTTP Listener, which communicates with the Listener Servlet. The Listener Servlet starts up ifweb90.exe for each new Oracle9*i* Forms module.

For more information about how Oracle9*i*AS Forms Services runs, see Oracle9iAS Forms Services in Action.

Deploying Your Application

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

1. Create your application in Oracle9*i* Forms Developer and save it.

   .fmb is a design time file that can only be opened in Oracle9*i* Forms Developer. .fmx is the runtime file created when you compile the .fmb and is used for Web deployment.

   For more information about Oracle9*i* Forms Developer, go to the Help menu in Oracle9*i* Forms Developer.

2. Create a configuration section in formsweb.cfg so that Oracle9*i* Forms can access your application module. You can do this at the end of formsweb.cfg in any open space. The syntax for creating a configuration file is as follows:

   [*my_application*]

   form=*name_of_module*

   For example,

   [application]

   form=hrapp.fmx

   In the previous example, Oracle9*i* Forms is running the application "application" and calling the file "hrapp.fmx".

   This means the Oracle9*i* Forms module hrapp.fmx will be callable on the Web by entering "...?config=application" in the Browser URL.

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

3. Make sure the .fmx file location is specified in the 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).

   Forms90_Path is an environment variable that can be set using an Oracle9*i* Forms environment file, an ASCII file referenced by envFile=<name of the file> in formsweb.cfg. The deault environment file used is default.env located in the Forms90\Server directory.

To create your own environment file copy default.env and modify it as needed by your application. In the application configuration section in formsweb.cfg, add the following parameter:

[application]

form=hrapp.fmx

envFile=<your file name>.env

4. 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 specified a configuration file, 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=application

Specifying Parameters

There are three ways to predefine paramter values for your Oracle9*i* Forms applications. You can define parameters by:

- **Editing your application settings in the default section of formsweb.cfg.**

  The default configuration section displays the default values that will be used by Oracle9*i* Forms. The default configuration section contains two parts: System Parameters and User Parameters. While you can edit the existing default configuration section, you cannot create new ones.

  For example, the default value of the system parameter that specifies how to execute the Forms applet under Microsoft Internet Explorer 5.x 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 formsweb.cfg file to read:

  IE=native

  You can override the system and user parameter values in the named application configuration section (see Creating configuration sections in formsweb.cfg.). For example:

```
[myApp]
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 formsweb.cfg 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.

- **Creating configuration sections in formsweb.cfg.**

  Under the configuration sections you created in step 2 of Deploying Your Application, you can specify parameters for your Oracle9*i* Forms applications. You can specify any application and system parameters that are in formsweb.cfg.

  For example, you can make the look and feel of the application to be the Oracle look and feel:

```
[Human Resources]
form=hrapp.fmx
envFile=<your file name>.env
userid=scott/tiger@orcl
```

lookandfeel= oracle

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:

[Human Resources]

form=hrapp.fmx

envFile=<your file name>.env

userid=scott/tiger@orcl

For other parameters you can edit, see Parameters in the formsweb.cfg File.

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

- **Editing the URL you use to access Oracle9*i* Forms applications.**

  You can directly type parameters into the URL that accesses your Oracle9*i* Forms 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.

Oracle9*i*AS Forms Services in Action

This sections describes how an Oracle9*i* Forms application runs in Oracle9*i*AS, 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 Oracle9*i*AS installation process.

When a user runs an Oracle9*i* Forms 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 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 Oracle9*i* Forms 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 substutions 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".

5. 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).

6. 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).

7. 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 Oracle9*i* Forms application (or if a newer version of Oracle9*i* Forms 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.

8. Once the Forms applet is running, it starts up a Forms session by contacting the Forms Listener Servlet at URL http://mymachine.com:7777/forms90/l90servlet.

9. The Oracle HTTP 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).

10. The Forms Listener Servlet (l90servlet) starts up a Forms runtime process (ifweb90.exe or f90webm) for the Forms session.

11. 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.

12. 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 formsweb.cfg file), 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 formsweb.cfg file (for example, form=c:\temp\myform).

13. The Forms sessions ends 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).

Client Browser Support

Users can view Oracle9*i* 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 Oracle9*i* Forms Developer menu and choose **Help | Forms on OTN...**

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".

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:

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.

4. 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).
5. Sends the HTML file to the user's browser.

3

Configuring Oracle9*i*AS Forms Services

Introduction

This chapter contains the following sections:

- Customizing Configuration Files
- Customizing Environment Variables and Registry Settings
- Creating Your Own Template HTML Files
- Including Graphics in Your Oracle9i Forms Application
- Deploying Icons and Images Used by Oracle9iAS Forms Services
- Using HTTPS with the Forms Listener Servlet
- Using the Hide User ID / Password Feature
- Using an Authenticating Proxy to Run Oracle9i Forms Applications
- Enabling Language Detection

Customizing Configuration Files

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

- formsweb.cfg
- base.htm, basejini.htm, basejpi.htm, and baseie.htm
- web.xml
- forms90.conf
- registry.dat

When a user first starts an Oracle9*i*AS 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 modify the configuration files as your needs change.

formsweb.cfg

For a description and the location of the Forms Servlet configuration file (formsweb.cfg), see Chapter 2, formsweb.cfg.

Parameters Naming Files

Make sure the EnvFile parameter specifies the physical path to the environment file that contains the environment variable settings you want to use.

The four baseHTML parameters should also be set to point to appropriate files. Typically, lines like the following should appear in the default section at the start of the formsweb.cfg file:

```
baseHTML=base.htm
baseHTMLie=baseie.htm
baseHTMLJinitiator=basejini.htm
baseHTMLjpi=basejpi.htm
envfile=default.env
```

All of these parameters give 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/servlet.

Creating specific named configurations in formsweb.cfg

You can create specific named configurations in the formsweb.cfg file. These configurations can be requested in the end-user's query string of the URL used to run a form.

Create special configurations by adding the name of the configuration in brackets at the end of the formsweb.cfg file. Then, specify the parameters (only those you want to change) for this special configuration.

For example, to create a configuration to run forms in a separate browser window with a "generic" look and feel, add the following code to the formsweb.cfg file:

```
[sepwin]
forms=<module>
separateFrame=True
lookandfeel=Generic
```

The end-user would type the following URL to launch a form that uses the "sepwin" configuration:

```
http://server:port/forms90/f90servlet?config=sepwin
```

You can also use Enterprise Manager to create named configuration sections (see Chapter 6, "Enterprise Manager and Oracle9i Forms").

See Default formsweb.cfg File for other examples of special configurations.

Parameters in the formsweb.cfg File

Parameter	Required / Optional	Parameter Value
System 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.		
baseHTML	optional	Physical path to HTML file that contains applet tags.
baseHTMLJInitiator	required	Physical path to HTML file that contains JInitiator tags.
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>baseie.htm. This is required when using the Internet Explorer native JVM.
HTML delimiter	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.

Parameter	Required / Optional	Parameter Value
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.
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.		
form	required	Specifies the name of the top level Forms module (fmx file) to run.
userid	optional	Login string. For example: scott/tiger@ORA8.
otherparams	optional	This setting specifies command line parameters to pass to the Forms runtime process in addtion to form and userid. Default is: debug=%debug% buffer_records=%buffer% debug_ messages=%debug_messages% array=%array% query_ only=%query_only% quiet=%quiet% render=%render% host=%host% port=%port%
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.		
pageTitle	optional	Oracle Forms Server
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.
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%">`		
serverURL	required	/forms90/l90servlet (see Chapter 1, Forms Listener Servlet)

Parameter	Required / Optional	Parameter Value
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.
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 virutal 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.
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 OracleHome\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.

Parameter	Required / Optional	Parameter Value
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.
Parameters for JInitiator		
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-0013-0001-0002-ABCDEFABCDEF
jinit_exename	required	Default is jinit.exe#Version=1.3.1.2
jinit_mimetype	required (Netscape only)	Default is application/x-jinit-applet;version=1.3.1.2
Parameters for Sun's Java Plug-in		
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
Enterprise Manager configuration parameters		

Parameter	Required / Optional	Parameter Value
em_mode	required	1 is to enable. 0 is to disable. Applies only to the HTML-based version of the Enterprise Manager. TRUE indicates that all Enterprise Manager information is available, including metrics and servlet status. FALSE indicates that only configuration information is available.
OID (Oracle Internet Directory) configuration parameters		
oid_formsid	required	Configured during the Oracle9*i*AS installation, so you do not need to change this.
oracle_home	required	Configured during the Oracle9*i*AS installation, so you do not need to change this.

Default formsweb.cfg File

The default formsweb.cfg file contains the following:

```
# formsweb.cfg - Forms Servlet default configuration file
# ------------------------------------------------------
# This file defines parameter values used by the FormsServlet (f90servlet)

# ********************************
# DEFAULT CONFIGURATIONS
# ********************************
#
# These are the default settings.  Any of them may be overridden in the
# Named Configurations section.  If they are not overridden, then the
# values here will be used.
# System Paremeters cannot be overridden in the URL.  User Parameters can.
#

#
# 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.
```

```
#

baseHTML=base.htm
baseHTMLjinitiator=basejini.htm
baseHTMLjpi=basejpi.htm
baseHTMLie=baseie.htm

HTMLdelimiter=%

#
# WorkingDirectory defaults to <oracle_home>/forms90 if unset.
#
workingDirectory=
envFile=default.env

#
# The next parameter specifies how to execute the Forms applet under
# Microsoft Internet Explorer 5.x.  Put IE=native if you want the
# Forms applet to run in the browser's native JVM.
#
IE=JInitiator

#
# 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)
#

#
# 1) Runform arguments:
#
form=test.fmx
userid=

#
# These settings support running and debugging a form from the Builder:
#
otherparams=debug=%debug% buffer_records=%buffer% debug_messages=%debug_
messages% array=%array% query_only=%query_only% quiet=%quiet% render=%render%
host=%host% port=%port% play=%play% record=%record%
debug=no
buffer=no
```

```
debug_messages=no
array=no
query_only=no
quiet=yes
render=no
host=
port=

#
# 2) HTML page title, attributes for the BODY tag, and HTML to add before and
#    after the form:
#
pageTitle=Oracle9iAS Forms Services
HTMLbodyAttrs=
HTMLbeforeForm=
HTMLafterForm=

#
# 3) Values for the Forms applet parameters:
#
serverURL=/forms90/l90servlet
codebase=/forms90/java
imageBase=DocumentBase
width=650
height=500
separateFrame=false
splashScreen=
background=
lookAndFeel=Oracle
colorScheme=teal
logo=

serverApp=default

#
# The following archive settings are for
#   archive_jini - settings for JInitiator
#   archive_ie   - settings for IE native JVM
#   archive      - settings for all other cases (Java Plugin, Appletviewer, etc)
#
archive_jini=f90all_jinit.jar
archive_ie=f90all.cab
archive=f90all.jar

#
```

```
# Number of times client should retry if a network failure occurs.  Only
# change after having read the documentation.
#
networkRetries=0

#
# 4) Parameters for JInitiator (used with Windows clients)
#

#
# Page displayed to Netscape users to allow them to download JInitiator.
# If you create your own page, you should set this parameter to point to it.
#
jinit_download_page=/forms90/jinitiator/us/jinit_download.htm

#
# Parameters related to the version of JInitiator.
#
jinit_classid=clsid:CAFECAFE-0013-0001-0003-ABCDEFABCDEF
jinit_exename=jinit.exe#Version=1,3,1,3
jinit_mimetype=application/x-jinit-applet;version=1.3.1.3

#
# 5) Parameters for the Java Plugin (used with non-Windows clients)
#

#
# Page displayed to users to allow them to download the JPI
# (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

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

#
# 6) EM config parameter
# Set this to "1" to enable Enterprise Manager to track Forms processes
#
```

```
em_mode=0

#
# 6) OID Config parameters (for Single Sign-On)
#
oid_formsid=%OID_FORMSID%
oracle_home=%ORACLE_HOME%

# ********************************
# NAMED CONFIGURATIONS
# ********************************
#
#  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".
#

#
# Example 1: configuration to run forms in a separate browser window with
#            "generic" look and feel (include "config=sepwin" in the URL)
#
[sepwin]
separateFrame=True
lookandfeel=Generic

#
# 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 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 4: configuration running the Forms ListenerServlet in debug mode
#            (debug messages will be written to the servlet engine's log file)
#
[debug]
serverURL=/servlet/l90servlet/debug
```

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

For a brief description and the locations of base.htm, basejini.htm, basejpi.htm, and baseie.htm, see Chapter 2, base.htm, basejini.htm, basejpi.htm, and baseie.htm.

Four baseHTML files are created for your system by the Oracle Universal Installer during Oracle9*i*AS 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 Oracle9*i* 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 formsweb.cfg, 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 formsweb.cfg 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 Oracle9*i* 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 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.

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, which are defined in Parameters in the formsweb.cfg File. All variables are replaced with the appropriate parameter values at runtime.

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, Oracle9*i*AS 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.

Default base.htm File

```
<HTML>
<!-- FILE: base.htm (Oracle Forms)                                      -->
<!--                                                                    -->
<!-- This is the default baseHTML 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="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="module=%form% userid=%userid% sso_userid=%sso_userid%
%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%">

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

%HTMLafterForm%

</BODY>
</HTML>
```

Default basejini.htm File

```
<HTML>
<!-- FILE: basejini.htm (Oracle Forms)                                    -->
<!--                                                                      -->
<!-- This is the default baseHTML 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="/forms90/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="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="module=%form% userid=%userid% sso_userid=%sso_userid%
%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%">
<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"

        networkRetries="%networkRetries%"
        serverArgs="module=%form% userid=%userid% sso_userid=%sso_userid%
```

```
%otherparams%"
        separateFrame="%separateFrame%"
        splashScreen="%splashScreen%"
        background="%background%"
        lookAndFeel="%lookAndFeel%"
        colorScheme="%colorScheme%"
        serverApp="%serverApp%"
        logo="%logo%"
        imageBase"="%imageBase%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

Default basejpi.htm File

```
<HTML>
<!-- FILE: basejpi.htm (Oracle Forms)                                    -->
<!--                                                                     -->
<!-- This is the default baseHTML 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="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="module=%form% userid=%userid% sso_userid=%sso_userid%
%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%">
<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"

        networkRetries="%networkRetries%"
        serverArgs="module=%form% userid=%userid% sso_userid=%sso_userid%
%otherparams%"
        separateFrame="%separateFrame%"
        splashScreen="%splashScreen%"
        background="%background%"
        lookAndFeel="%lookAndFeel%"
        colorScheme="%colorScheme%"
```

```
        serverApp="%serverApp%"
        logo="%logo%"
        imageBase"="%imageBase%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

Default baseie.htm File

```
<HTML>
<!-- FILE: baseie.htm (Oracle Forms)                                    -->
<!--                                                                    -->
<!-- This is the default baseHTML 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="networkRetries" VALUE="%networkRetries%">
<PARAM NAME="serverArgs"
       VALUE="module=%form% userid=%userid% sso_userid=%sso_userid%
%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%">

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

%HTMLafterForm%

</BODY>
</HTML>
```

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 Oracle9*i* Forms in your Oracle9*i*DS or Oracle9*i*AS 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 adminstration 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 Oracle9*i* Forms using static HTML pages (rather than the Forms Servlet).

When Oracle9*i* 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).

Default web.xml File

```
- <web-app>
  <display-name>Forms 9i Services</display-name>
  <description>Oracle 9iAS: Forms 9i Services</description>
- <welcome-file-list>
  <welcome-file>l90servlet</welcome-file>
  </welcome-file-list>
- <!--  Forms 9i 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 9i listener servlet
  -->
- <servlet>
  <servlet-name>l90servlet</servlet-name>
  <servlet-class>oracle.forms.servlet.ListenerServlet</servlet-class>
  </servlet>
- <!--
 Forms 9i 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>
```

forms90.conf

For a description and the location of forms90.conf, see Chapter 2, forms90.conf.

The following table describes the virtual paths and servlet mappings:

Table 3–1 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>/forms90/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

Default forms90.conf

```
# $Id: forms90.conf,v 1.6 2001/10/26 00:52:16 cbarrow Exp $
# Name
#   forms90.conf
# Purpose
#   Apache mod_oc4j and mod_jserv configuration file for Forms 9i 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/(..*) "%FORMS_ORACLE_HOME%/forms90/java/$1"
```

```
  # Virtual path for JInitiator downloadable executable and download page
AliasMatch ^/forms90/jinitiator/(..*) "%FORMS_ORACLE_HOME%/jinit/$1"

  # Virtual path for runform.htm (used to run a form for testing purposes)
AliasMatch ^/forms90/html/(..*) "%FORMS_ORACLE_HOME%/forms90/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              ProductGroup2
    Oc4jMount /forms90/f90servlet   ProductGroup2
```

```
    Oc4jMount /forms90/f90servlet/* ProductGroup2
    Oc4jMount /forms90/l90servlet   ProductGroup2
    Oc4jMount /forms90/l90servlet/* ProductGroup2
</IfModule>
```

registry.dat

For a description and the location of registry.dat, see Chapter 2, registry.dat.

The main reason you would want to edit this file is to change the icon settings (see Icons). You can also change the default font and font settings by changing the following section in the registry.dat file:

```
default.fontMap.defaultFontname=Dialog
default.fontMap.defaultSize=900
default.fontMap.defaultStyle=PLAIN
default.fontMap.defaultWeight=PLAIN
```

Change any of the settings above to reflect your desired font setting. For example, if you want to change your default font to Times New Roman, replace **Dialog** with **Times New Roman**.

You can change the default font face mappings:

```
default.fontMap.appFontnames=Courier
New,Courier,courier,System,Terminal,Fixed,Fixedsys,Times,Times New Roman,MS Sans
Serif,Arial
default.fontMap.javaFontnames=MonoSpaced,MonoSpaced,MonoSpaced,Dialog,MonoSpaced,Dialog,Dialo
g,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 Default registry.dat.

Default registry.dat

```
#

# This is the Registry file.
#
# This file contains the logical [Java] Class name and an associated
# [numerical] identifier that will be used to refer to objects of the
# class in order to reduce the amount of information that needs to be
# repeatedly transmitted to the client.
```

```
#
# This file is of the Form understood by java.util.Properties (for now)
#
# The System Level sound file is relative to the CODEBASE
#
#
oracle.classById.1=oracle.forms.engine.Runform
oracle.classById.4=oracle.forms.handler.FormWindow
oracle.classById.5=oracle.forms.handler.AlertDialog
oracle.classById.6=oracle.forms.handler.DisplayList
oracle.classById.7=oracle.forms.handler.LogonDialog
oracle.classById.8=oracle.forms.handler.DisplayErrorDialog
oracle.classById.9=oracle.forms.handler.ListValuesDialog
oracle.classById.10=oracle.forms.handler.EditorDialog
oracle.classById.11=oracle.forms.handler.HelpDialog
oracle.classById.12=oracle.forms.handler.FormStatusBar
oracle.classById.13=oracle.forms.handler.MenuInfo
# oracle.classById.14=UNUSED
oracle.classById.15=oracle.forms.handler.ApplicationTimer
oracle.classById.16=oracle.forms.handler.MenuParametersDialog
oracle.classById.17=oracle.forms.handler.PromptListItem
oracle.classById.18=oracle.forms.handler.CancelQueryDialog
oracle.classById.257=oracle.forms.handler.TextFieldItem
oracle.classById.258=oracle.forms.handler.TextAreaItem
oracle.classById.259=oracle.forms.handler.FormCanvas
oracle.classById.261=oracle.forms.handler.ButtonItem
oracle.classById.262=oracle.forms.handler.CheckboxItem
oracle.classById.263=oracle.forms.handler.PopListItem
oracle.classById.264=oracle.forms.handler.TListItem
oracle.classById.265=oracle.forms.handler.CfmVBX
oracle.classById.266=oracle.forms.handler.CfmOLE
oracle.classById.267=oracle.forms.handler.RadioButtonItem
oracle.classById.268=oracle.forms.handler.ImageItem
oracle.classById.269=oracle.forms.handler.IconicButtonItem
oracle.classById.270=oracle.forms.handler.BlockScroller
oracle.classById.271=oracle.forms.handler.JavaContainer
oracle.classById.272=oracle.forms.handler.TabControl
oracle.classById.273=oracle.forms.handler.ComboBoxItem
oracle.classById.274=oracle.forms.handler.TreeItem
oracle.classById.281=oracle.forms.handler.PopupHelpItem

#
# Defaults for the Font details, all names are Java Font names.  Each of
# these parameters represents the default property to use when none is
# specified.
```

```
#
# defaultFontname represents the default Java fontName.
# defaultSize     represents the default fontSize.  Note that the size is
#                 multiplied by 100 (e.g. a 10pt font has a size of 1000).
# defaultStyle    represents the default fontStyle, PLAIN or ITALIC.
# defaultWeight   represents the default fontWeight, PLAIN or BOLD.
#
default.fontMap.defaultFontname=Dialog
default.fontMap.defaultSize=900
default.fontMap.defaultStyle=PLAIN
default.fontMap.defaultWeight=PLAIN

#
# Default Font Face mapping.
#
# appFontname  represents a comma delimited list of Application Font Names.
# javaFontname represents a comma delimited list of Java Font Names.
#
# The number of entries in the appFontname list should match the number in
# the javaFontname list.  The elements of the list are comma separated and
# *all* characters are taken literally, leading and trailing spaces are
# stripped from Face names.
#
# Note that this file uses the Java 1.1 Font names in order to be able to
# handle the NLS Plane (BUG #431051)
#
default.fontMap.appFontnames=Courier
New,Courier,courier,System,Terminal,Fixed,Fixedsys,Times,Times New Roman,MS Sans
Serif,Arial
default.fontMap.javaFontnames=MonoSpaced,MonoSpaced,MonoSpaced,Dialog,MonoSpaced
,Dialog,Dialog,Serif,Serif,Dialog,SansSerif

#
# The Application Level icon files are relative to the DOCUMENTBASE
#   example: icons/
# or an absolute URL.
#   example: http://www.forms.net/~luser/d2k_project/
#
default.icons.iconpath=
default.icons.iconextension=gif
#
# Application level settings to control UI features
#
app.ui.lovButtons=false
app.ui.requiredFieldVA=false
```

```
#    The background color is specified as an RGB triple.
app.ui.requiredFieldVABGColor=255,0,0
```

Customizing Environment Variables and Registry Settings

The environment variables for Forms runtime are required by the ifweb90.exe executable. 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.

default.env

The environment variables required for a Forms runtime process (for example, PATH, ORACLE_HOME, FORMS90_PATH) can be defined in an environment file. Any environment variables not defined in that file are inherited from the servlet engine (OC4J). The environment file must be named in the envFile parameter in formsweb.cfg.

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).

Important environment variables specified in default.env.

Envirionment 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.

Envirionment Variable	Valid Values	Purpose
FORM90_PATH	%ORACLE_HOME%\forms90 (default)	Specifies the path that Oracle9*i* 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* (:).
NLS_LANG	Defaults to language, territory, and charset specified by the database. For more information about NLS_LANG, see your Oracle Server Reference Manual.	Specifies language-dependent operation of applications (for example, French, Japanese, or German). NLS_LANG has three components: ***language*** - specifies supported languages used for conventions such as Oracle messages, day names, and month names. The language argument specifies default values for the territory or charset arguments, so either (or both) *territory* or *charset* can be omitted. *territory* - Specifies territory-dependent operation of applications (for example, America, Brazil, or Canada). *charset* - Specifies the character set used by the client application (for example, US7ASCII, JA16EUC. For more information about NLS_LANG, see your Oracle Server Reference Manual.
DE_PREFS_TABSIZE	Any positive integer	Defines the size of a Tab (in characters) in the PL/SQL editor. Default value is 2
FORMS90_ CATCHTERM	FALSE, TRUE (default)	Bypasses the normal Oracle9*i*AS Forms Services crash handling, and enables an operating system crash file (e.g. a Core Dump on Unix) to be generated. Normally this variable should not be set unless requested by Oracle Support.

Envirionment Variable	Valid Values	Purpose
FORMS90_TRACE_ PATH	Valid writable O/S Path	The FORMS90_TRACE_PATH environment variable specifies the location of dump files produced as the result of a crash of any of the Oracle9*i*AS Forms Services executables. The dump files contain diagnostic information about events at the time the process crashed. If FORMS90_TRACE_PATH is not set and a crash occurs, Oracle9*i*AS Forms Services will attempt to place the dump file in the directory from which the executable was called. Set this environment variable in the jserv.properties file or the default.env file.
FORMS90_MMAP	FALSE, TRUE (default)	FORMS90_MMAP configures Oracle9*i*AS Forms Services to use memory mapped file i/o when running Oracle9*i* Forms applications. This reduces the amount of memory required on a system when more than one user is running the same Form. By default FORMS90_MMAP is set on. Switching Memory mapping to off (FALSE) may be desirable in a development environment as memory mapped file i/o has the side effect of locking the forms in use preventing a developers from replacing them with a new version. However, be aware that doing so may increase memory usage and reduce performance. For production systems, FORMS90_MMAP should be unset or set to a value of TRUE
FORMS90_REJECT_ GO_DISABLED_ITEM	FALSE, TRUE (default)	In Oracle9*i* Forms the Go_Item() built-in is not allowed to navigate the cursor to an item that is currently disabled. Setting this variable to False will restore the behavior found in earlier versions of Oracle9*i* Forms where this was possible. This variable should only be used for backwards compatibility and may be removed in a future version of the product.

Envirionment Variable	Valid Values	Purpose
FORMS90_SWITCH_ JAVA_EVENTS	FALSE (default), TRUE	This variable is used to change the way in which Oracle9*i* Forms queues up and handles events from embedded JavaBeans. Normally this variable should not be set unless requested by Oracle Support
FORMS90_ SEPARATE_ DEBUGGER	FALSE (default), TRUE	This variable effects the way that the Forms Builder displays the windows created by the Forms Debugger. Setting this variable to TRUE will create the Debugger windows externally from the main Forms MDI interface. Use this setting if you need to use screen reader software such as JAWS with the debugger.
FORMS90_CLAF	<NULL> (default), TRUE	Setting FORMS90_CLAF to TRUE will restore the Forms Builder User interface to the "Classic" (6i) Forms look and feel. This may be suitable for users who require high contrast color schemes

Note: On NT, Oracle9*i* Forms reads Oracle environment settings from the registry unless they are set as environment variables.

Default default.env File for Windows

```
# default.env - default Forms environment file, Windows version
#
# This file is used to set the Forms runtime environment parameters.
# If a parameter is not defined here, the value in the Windows registry
# will be used. If no value is found in the registry, the value used will
# be that defined in the environment in which the servlet engine (OC4J
# or JServ) was started.
#
# NOTES
# 1/ The Forms installation process should replace all occurrences of
#    <percent>FORMS_ORACLE_HOME<percent> with the correct ORACLE_HOME
#    setting, and all occurrences of <percent>O_JDK_HOME<percent> with
#    the location of the JDK (usually $ORACLE_HOME/jdk).
#    Please make these changes manually if not.
```

```
# 2/ Some of the variables below may need to be changed to suite your needs.
#    Please refer to the Forms documentation for details.
#
ORACLE_HOME=%FORMS_ORACLE_HOME%

  #
  # Search path for Forms applications (.fmx files, PL/SQL libraries)
  # If you need to include more than one directory, they should be semi-colon
  # separated (e.g. /private/dir1;/private/dir2)
  #
  # FORMS90_PATH=%FORMS_ORACLE_HOME%/forms90

  #
  # The PATH setting is not required, if the Forms executables and dll's are
  # in %ORACLE_HOME%\bin:
  #
  # PATH=%FORMS_ORACLE_HOME%\bin

  #
  # 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 9i
    # oracle_home value to use Graphics applications.
    #
    #FORMS90_ORACLE9i_HOME=<your Graphics 9i oracle_home here>

    #
    # Search path for Graphics applications
    #
    #GRAPHICS90_PATH=

  #
  # System settings
  # ---------------
  # You should not normally need to modify these settings
  #
  ORA_NLS33=%FORMS_ORACLE_HOME%\ocommon\ADMIN\DATA

  FORMS90=%FORMS_ORACLE_HOME%\forms90
```

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=%FORMS_ORACLE_HOME%

  #
  # Search path for Forms applications (.fmx files, PL/SQL libraries)
  #
  FORMS90_PATH=%FORMS_ORACLE_HOME%/forms90

  #
  # Java class path
  #   This is required for the Forms debugger
  #   You can append your own Java code here)
  #
  CLASSPATH=%FORMS_ORACLE_HOME%/forms90/java/debugger.jar:%FORMS_ORACLE_
HOME%/forms90/java/utj90.jar:%FORMS_ORACLE_HOME%/jlib/ewt3.jar:%FORMS_ORACLE_
HOME%/jlib/share.jar

  #
  # The PATH setting is not required if the Forms executables are
  # in <ORACLE_HOME>/bin:
  #
  # PATH=%FORMS_ORACLE_HOME%/bin

  #
  # 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.
    #
    #FORMS90_ORACLE6i_HOME=<your Graphics 6i oracle_home here>

    #
    # Search path for Graphics applications
    #
    GRAPHICS60_PATH=

  #
  # 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=%FORMS_ORACLE_HOME%/lib:%O_JDK_HOME%/jre/lib/sparc:%O_JDK_
HOME%/jre/lib/sparc/native_threads:%LD_LIBRARY_PATH%
```

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.

Including Graphics in Your Oracle9*i* Forms Application

In order to integrate graphics applications with your Oracle9*i* Forms applications, you must set the path definition in the Forms Servlet environment to include graphics as follows:

PATH=%FORMS9I_HOME%\bin;%ORACLE_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 *Oracle9i Forms Developer and Forms Services: Migrating Forms Applications from Forms6i*.

Deploying Icons and Images Used by Oracle9*i*AS Forms Services

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

Icons

When deploying an Oracle9*i*AS Forms Services 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 formsweb.cfg configuration file. Alternatively, you can change the default.icons.iconpath value of the registry.dat file in the forms90/java/oracle/forms/regsitry directory.

Setting the imageBase parameter to codebase enables Oracle9*i* 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 Oracle9*i* Forms installation.

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 Development Kit (JDK).

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 Oracle9*i* 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 Oracle9*i* Forms default icons (for example, icons present in the default smart icon bar), as they are part of the f90all.jsr file,

---

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)

4. Modify the iconextension parameter:

   default.icons.iconextension=gif
   or
   default.icons.iconextension=jpg

To reference the application file:

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

For example:

```
[my_app]
ServerApp=/appfile/myapp
```
(for an absolute path)

or

```
[my_app]
ServerApp=appfile/myapp
```
(for a relative path, relative to the CodeBase directory)

Table 3–2 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 Oracle9*i* 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.

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 in the formsweb.cfg file:

```
<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.

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.

Creating a JAR File

The SunSoft JDK comes with an executable called *jar*. This utility enables you to store files inside a Java archive. For more information, see www.java.sun.com.

For example:

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

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

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

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 formsweb.cfg file or HTML file (for the images).
- What you specify in the imageBase parameter in your formsweb.cfg 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.

DocumentBase

The default search path is relative to the DocumentBase. In this case, you do not need to specify the imageBase parameter:

Table 3–3 Search Paths for Icons and Images

	Location specified	Search path used by Forms Services
Icons	default	http://host/documentbase

Table 3–3 Search Paths for Icons and Images

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

CodeBase

Use the imageBase=CodeBase parameter in the formsweb.cfg file to enable the search of the icons and images in a JAR file:

Table 3–4 Icons and Images Search Paths Used by Oracle9iAS Forms Services

	Location specified	Search path used by Forms Services
Icons	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
Images	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 3–4 Icons and Images Search Paths Used by Oracle9iAS 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.

Using HTTPS with the Forms Listener Servlet

Using HTTPS with Oracle 9*i* Forms is no different than using HTTPS with any other Web-based application.

Server Requirements

HTTPS requires the use of digital certificates. Because Oracle9*i*AS Forms Services servlets are accessed via your Web server, you do not need to purchase special certificates for communications between the Oracle9*i* Forms client and the server. You only need to purchase a certificate for your Web server from a recognized Certificate Authority.

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 using the test certificate supplied with Oracle9*i*AS for test purposes, you must edit the JInitiator certdb.txt file and append the contents of the demo root certificate, which is located in <9iAS oracle_home/Apache/Apache/conf/ssl.crt/demoCAcert.txt. Otherwise, you will get the following error when attempting to run a form: javax.net.ssl.SSLException: SSL handshake failed:X509CertChainInvalidErr.

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

Using the Hide User ID/Password Feature

With Oracle9*i*AS Forms Services, the userid parameter value is not included in the HTML generated by the Forms Servlet.

By default, this feature enables Oracle9*i*AS 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 Oracle9*i* Forms is installed. They contain syntax like "userid=%userid%".
    - Use the Forms Servlet rather than static HTML files.

Using an Authenticating Proxy to Run Oracle9*i* Forms Applications

The default configuration as set up by the Oracle9*i*AS 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 Oracle9*i*AS installation process are /forms90/java and /forms90/l90servlet. As these are under the document base of the page (/forms90), authenticating proxies will work.

Enabling Language Detection

Oracle9*i* 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 Oracle9*i* 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.

To specify language detection:

For each configuration section in formsweb.cfg, 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
```

How Language Detection Works

When the Forms Servlet receives a request for a particular configuration (for example, http://myserv/servlet/f60servlet?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/f60servlet), it will look for a language-specific section in the default section matching the first language (for example, [.fr]).

Multi-Level Inheritance

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

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.

4

Using Oracle9*i*AS Forms Services with the HTTP Listener and OC4J

Introduction

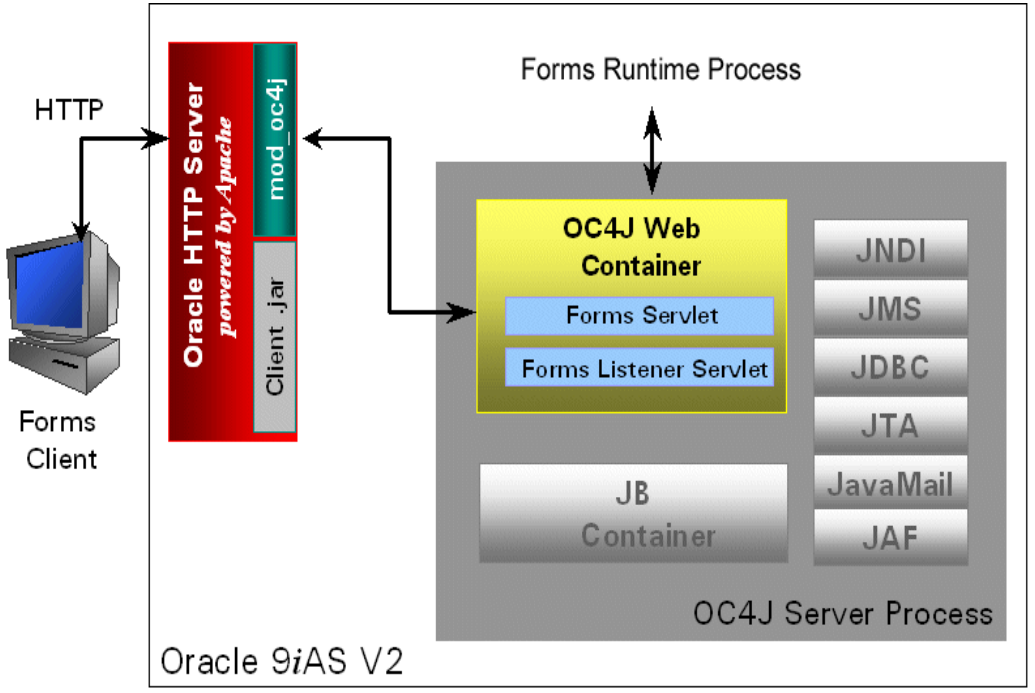
Oracle9*i*AS Containers for J2EE (OC4J) is a complete J2EE (Java 2 Platform Enterprise Edition) server written entirely in Java that executes in a standard Java Runtime Environment (JRE). It provides a complete J2EE environment that contains, among other things, an OC4J Web container.

This chapter contains the following sections:

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 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 Oracle9*i* Forms sessions, the requests are received by the HTTP 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:



Performance/Scalability Tuning

The steps for tuning the Forms Listener Servlet are similar to steps for tuning any high throughput servlet application.

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).

Set the maxClient directive to a High value

It is best to let the HTTP Listener determine when to create more HTTPD daemons. Therefore, set the maxClient 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.

MaxClient=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.

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 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 *Oracle9i Application Server Performance Guide* (available on Oracle9*i*AS Disk 1 CD or OTN at http://otn.oracle.com/docs/products/ias/content.html).

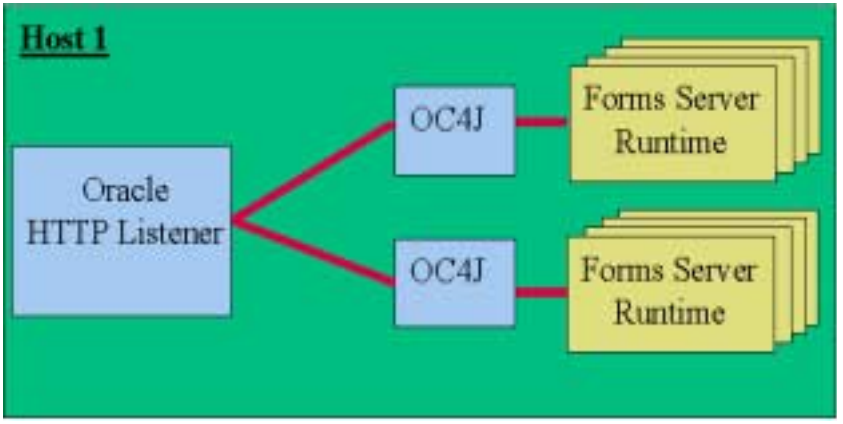
For more Forms-specific information, see the *Oracle9i Forms Developer and Forms Services* release notes.

In this section we take a look at four scenarios:

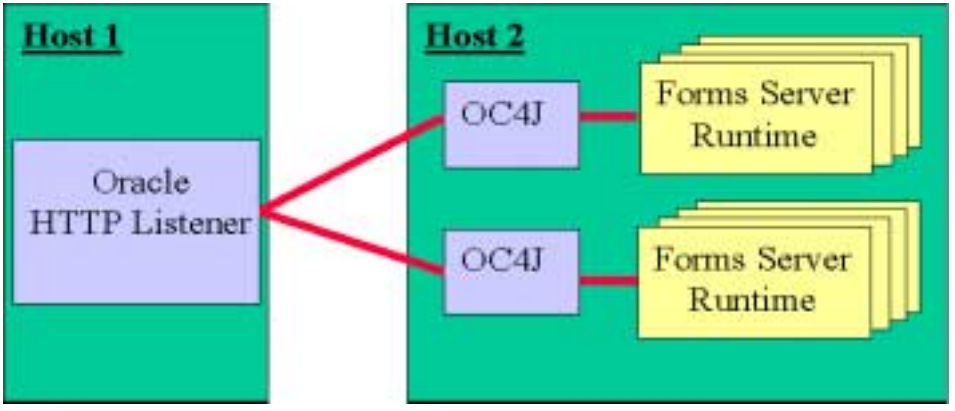
- How to balance incoming requests between multiple OC4J engines on the same host as the Oracle HTTP Listener.
- How to balance incoming requests between multiple OC4J engines on a different host to the Oracle HTTP Listener.

- How to balance incoming requests between multiple OC4J engines on multiple different hosts and multiple different hosts each running an Oracle HTTP Listener.
- How to balance incoming requests between multiple OC4J engines on a single host but with multiple different hosts each running an Oracle HTTP Listener.

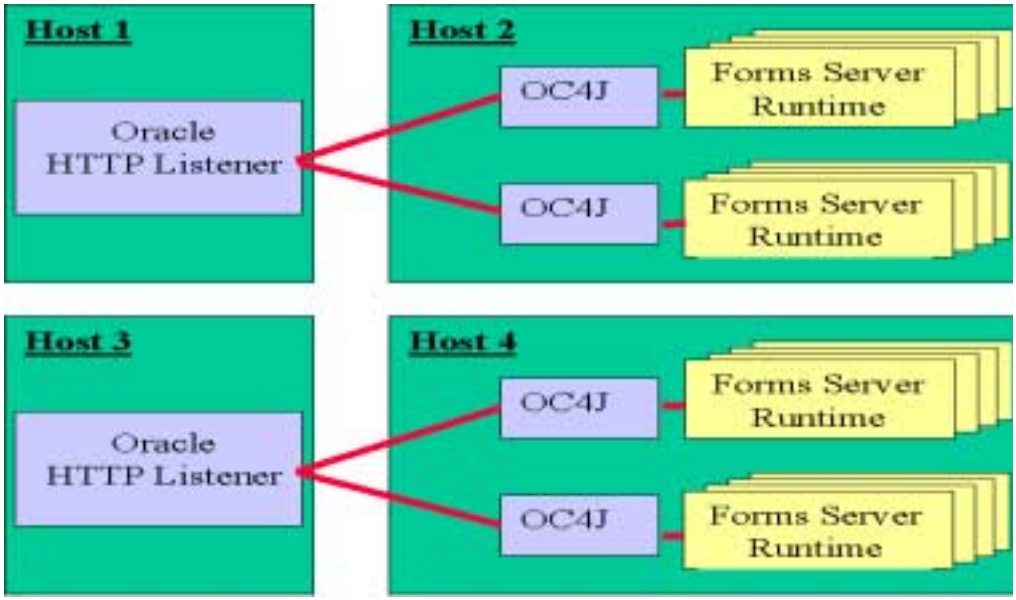
Case 1: Multiple OC4J engines on the same host as the Oracle HTTP Listener.



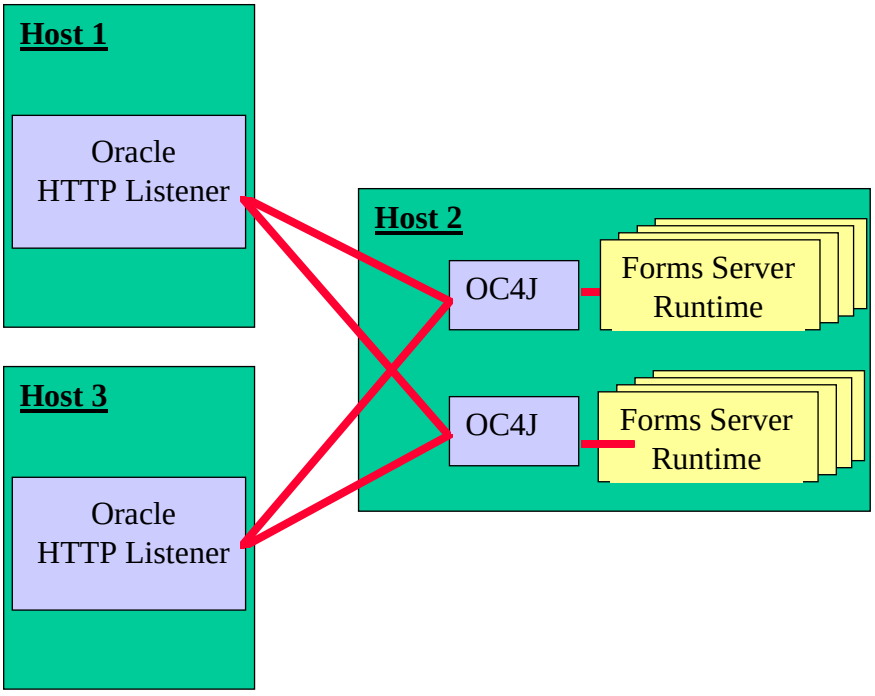
Case 2: Multiple OC4J engines on a different host to the Oracle HTTP Listener.



Case 3: Multiple OC4J engines and multiple Oracle HTTP Listeners on different hosts.



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



5

Using Oracle9*i*AS Forms Services with SSO and OID

Introduction

Oracle9*i*AS Forms Services applications can be run in a single sign-on (SSO) environment using Oracle Login Server (Single Sign-On Server) and Oracle Internet Directory (OID) to store user name and password information. SSO is designed to work in a portal environment such as that provided by Oracle9*i* Application Server, where multiple Web-based applications are accessible through the portal . 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 insecure and expensive.

You can also use

This chapter contains the following sections:

Single Sign-On (SSO)

A detailed description of SSO is available from the *Oracle9i Application Server Security Guide* (Chapter 6 - "Configuring Oracle9*i*AS Single Sign-On"). This section will explore the features which are relevant to Oracle9*i* Forms development.

To use single sign-on support:

1. The URL must be protected. Creating alias names for the Forms Listener Servlet will allow you to use SSO for selected applications. Protect the Forms URL by adding the following section to the mod_osso.conf file:

```
<Location forms90/f90servlet>
   require valid-user
   authType Basic
  </Location>
```

   The default Forms URL is "forms90/f90servlet".

2. Create users using OID, Delegated Administration Services (DAS), and assign/create resource information to these users. The resource information contains the DB credentials, and the resource name must match the name of the named user configurations for the SSO-enabled application defined in the formsweb.cfg file. The resource name is passed with the Oracle9*i*AS Forms Services URL as the value of the "config" query parameter.

3. Create a configuration section corresponding to the resource name used for Oracle9*i* Forms applications. For example:

```
...
[customers]
form=customers.fmx
...
[orders]
form=orders.fmx
...
```

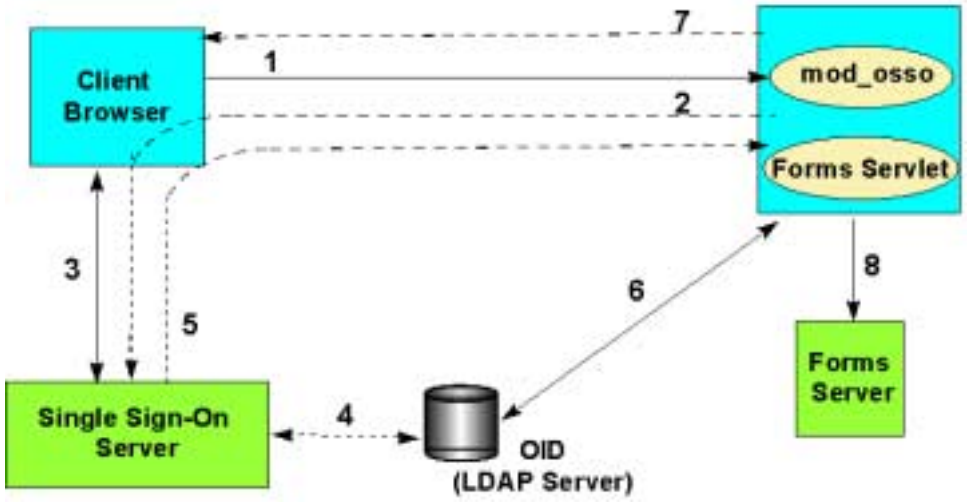
4. Choose a resource name. For example, the URL might be
   http://<server>:<port>/forms90/f90servlet?config=customers
   This will cause the request to be authenticated as shown below.

   If a different configuration parameter (for example, http://.../forms90/f90servlet?config=orders) is used, the DB credentials configured for this user with resource name "orders" will be used by the Forms Server to login to the DB.

5. Session cookies are used to track SSO logins - these are only available to the SSO server and are lost when the browser session ends.

Authentication Flow

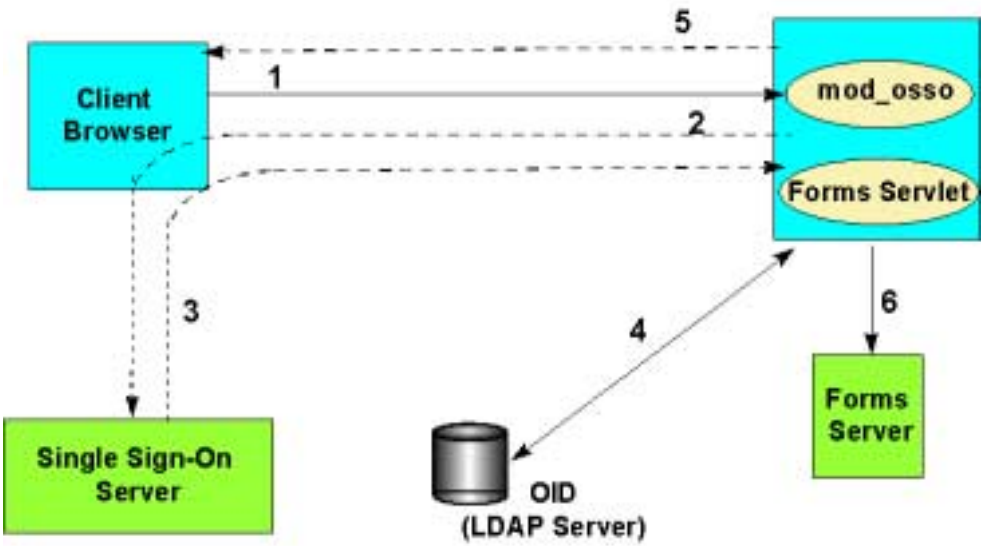
The following is the authentication flow of SSO support in Oracle9*i* Forms the first time the user requests an Oracle9*i*AS Forms Services URL:



1. The user requests a Forms URL.
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 userid 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 SSO support in Oracle9*i* Forms when a user, authenticated through another Partner Application, requests an Oracle9*i*AS Forms Services URL.

a



1. The user requests Forms URL.
2. Forms Servlet redirects the user to the SSO server.
3. The user gets redirected to the URL with sso_userid information.
4. Forms Servlet gets the database credentials from OID.
5. Forms Servlet sets the userid parameter in the Runform session and the applet connects to the Forms Listener Servlet.
6. Forms Servlet starts the Forms Server.

Single Sign-On with Some Applications, Not Others

Oracle 9*i*AS Forms Services allows you to run both single sign-on and publicy viewed applications simultaneously. In order to enable this functionality, you need to add a second alias name for the Forms Servlet. This second Forms Servlet name is required to build a second Forms Services instance.

For security reason it is not recommended you have two Forms Servlet names sharing the same formsweb.cfg file when one of them is registered with SSO. For example: If applicationA is protected by Single Sign-On authentication while applicationB is not, then it must be guaranteed that applicationA cannot be run by

the second Forms Services instance , which is not using SSO. The only way to distinguish SSO-protected applications from those being publicly available is to use separate configuration files. ApplicationA will not share the same formsweb.cfg file with applicationB.

The following steps will allow you to run single sign-on and publicly viewed applications:

- Create a second stand alone server instance
- Add an alias name for the Forms Servlet
- Register the Oracle9i Forms alias name with mod_oc4j

Create a second stand alone server instance

1.Add another alias name for the Forms Servlet to the OC4J web.xml file.

2.Register the aliased Forms Servlet with mod_oc4j.

Add an alias name for the Forms Servlet

Navigate to the \applications\forms90app\forms90web\WEB-INF\ directory of your Oracle9*i*AS Forms Services OC4J installation. Open the web.xml configuration file in a text editor and add the following lines to the appropriate location:

```
<servlet>
<servlet-name>f90</servlet-name>
      <servlet-class>oracle.forms.servlet.FormsServlet</servlet-class>
            <init-param>
           <param-name>configFileName</param-name>
           <param-value><your configuration file name goes
here></param-value>
        </init-param>
</servlet>
<servlet-mapping>
      <servlet-name>f90</servlet-name>
      <url-pattern>/f90*</url-pattern>
 </servlet-mapping>
```

In this example the new Forms Servlet name is “f90”, but you can give it any name. The servlet initialization parameter configFileName takes the name and location of the formsweb.cfg file to be used with this servlet. The best way to create the new configuration file is to copy the formsweb.cfg file from the forms90/server directory and to rename the copy. Edit the copied configuration file and remove all the application configurations that should not be accessible without SSO. Replace

"<your configuration file name goes here>" in above example with the location and the name of the formsweb.cfg copy you created.

Register the Oracle9*i* Forms alias name with mod_oc4j

For the new Forms Servlet name to be recognized by Oracle9*i*AS and served by OC4J, you need to register the servlet with mod_oc4j. This registration is done using the forms90.conf file located in the forms90/server directory of your Oracle9*i*AS Forms Services installation.

The following entry for the f90servlet and l90servlet alias names (in regular type) is at the end forms90.conf. Add the lines in bold type:

```
# Config. for OC4J
<IfModule mod_oc4j.c>
   Oc4jMount /forms90          ProductGroup2
   Oc4jMount /forms90/f90servlet   ProductGroup2
   Oc4jMount /forms90/f90servlet/* ProductGroup2
   Oc4jMount /forms90/f90     ProductGroup2
   Oc4jMount /forms90/f90/*  ProductGroup2
   Oc4jMount /forms90/l90servlet   ProductGroup2
   Oc4jMount /forms90/l90servlet/* ProductGroup2
</IfModule>
```

After stopping and restarting the Oracle HTTP Server, the Forms Servlet will be accessible by using:

http://<hostname>:<port>/forms90/f90servlet and
http://<hostname>:<port>/forms90/f90.

The difference is that the forms90/f90servlet root might be protected by single sign-on and the forms90/f90 path uses a different Forms configuration file (other than formsweb.cfg).

Note: The Forms Listener Servlet "l90servlet" can be used with both configurations, so there is no need to create an alias name for this servlet too. The Forms Listener Servlet does not directly read from the formsweb.cfg file and thus, security is not affected when using the same Listener Servlet for both instances.

You can test the configuration by typing

http://<hostname>:<port>/forms90/f90/admin which should bring up the Listener Servlet test page.

6

Enterprise Manager and Oracle9*i* Forms

There are two Enterprise Manager user interfaces that are shipped with Oracle9*i*AS:

- An HTML-based tool that you launch from your default browser
- A Java-based console that you launch from the command line

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

For Oracle9*i*AS Forms Services, use the HTML-based Enterprise Manager to:

- Monitor metrics for an Oracle9*i*AS Forms Services instance
- Monitor metrics for user sessions
- Terminate user sessions
- Configure parameters for an Oracle9*i*AS Forms Services instance

For Oracle9*i*AS Forms Services, use the Java-based Enterprise Manager console to:

- Access metrics for multiple Oracle9*i*AS Forms Services instances
- Monitor CPU usage and memory usage events for Oracle9*i*AS Forms Services instances

> **Note:** For more information on Oracle Enterprise Manager, see *Oracle9i Application Server Administrator's Guide*.

Also, online help is available when you launch Oracle Enterprise Manager.

7

Tracing and Diagnostics

When you develop and deploy Oracle9*i* Forms applications, it is helpful to have information that allows you to optimize your applications. Tracing and diagnostic tools that are available with Oracle9*i* Forms allow you to analyze the performance and resource consumption of your Oracle9*i* Forms applications at runtime. You can use trace output to diagnose performance and other problems with Oracle9*i* Forms applications.

The following tools are available to collect trace information for Oracle9*i* 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 Oracle9*i* 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 Oracle9*i* Forms sessions, monitor Oracle9*i* Forms-related network traffic, and debug site configuration problems.
- Oracle Trace: A tool to gather and analyze the performance of the Oracle Database, Oracle9*i*AS Forms Services, and other products which implement the Oracle Trace API for tracing.

This chapter describes the tracing tools and contains the following sections:

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:

Configuring Forms Trace

You define the events that you want to trace in the ftrace.cfg file or in the URL when you start the trace. An event is something that happens inside Oracle9*i* Forms as a direct or indirect result of a user action. See "List of Traceable Events" for a list of events and their corresponding event numbers.

ftrace.cfg

The ftrace.cfg file is installed in the %forms_home%/forms90/server directory by default, and the FORMS90_TRACE_PATH environment variable is set in the default.env file. (The FORMS90_TRACE_PATH environment variable specifies the location of the ftrace.cfg file and the location of trace output files.)

In a text editor, edit the ftrace.cfg configuration file to specify named event sets. An event set specifies a set of events that you can trace simply by specifying the event set name rather than each event number individually when you start the trace.

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

```
//
// example ftrace.cfg file
// This file is used to specify event groups for use with Forms Trace
//
// The file format is
// name1: event1, event2, ... , event_n
// name2: event1-event_m
```

```
//
all: 0-199
errors: 0-3
custom1: 32-46, 65, 66, 96, 194
```

Note the following:

- There must be a blank line between keyword entries.
- Keywords can be 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.

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"

URL Parameter Options

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

Record = forms | otrace

Tracegroup =

Log = <filename>

Table 7–1 Forms Trace Command Line Parameters

Parameter	Values	Description
Record	forms	Enables Forms Trace.
	otrace	Enables Oracle Trace integration

Table 7–1 Forms Trace Command Line Parameters

Parameter	Values	Description

Tracegroup		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		Specifies where trace information is saved. If a directory is not specified, the file is saved in the directory specified by the FORMS90_TRACE_PATH environment variable. (If this variable is not set, the output 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.

Starting the Trace

You start a trace by specifying trace entries in the URL or formsweb.cfg file. Entries should include the grouping of events to collect and the trace file name. Trace collection starts when the form executes.

The following are sample URLs to start a trace:

```
http://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.
log
```

A later release of Oracle9*i* Forms will implement a method for starting a trace via a built-in. The most recent information regarding Oracle9*i* Forms, including updated documentation, whitepapers, and viewlet demonstrations, is available on OTN at http://otn.oracle.com.

Viewing Forms Trace Output

Trace data is stored in a binary file with a *.trc extension. To view trace data, you must either:

- Use the Upload/Translate utility to convert the data in the *.trc file to XML format, and then view the data using an XML viewer.
- Use the Upload/Translate utility to upload the data in the *.trc file to database tables. (See "Creating Database Tables for the Trace Data" for information about scripts that will create the tables for you.)

Running the Upload/Translate Utility

The Upload/Translate utility performs two functions:

- Converts trace data to XML format.
- Uploads trace data to database tables.

To convert trace data to XML format, on the command line, type

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

to create myfile.xml.

To upload trace data to database tables, on the command line, type

```
java oracle.forms.diagnostics.Xlate datafile=a.trc userid=[password]=pw
url=[server]:1521:orcl9i
```

to upload the trace data to the tables.

The following are valid command line options for the utility:

Table 7–2 Translate Utility Command Line Options

Parameter	Description
DataFile	Specifies the name of the binary trace data collection file.
XMLFile	Specifies the name of the XML output file, if required.
Url	Specifies the database connection of the form -- hostname:port:oracle_sid.
UserName	Specifies the database username.
Password	Specifies the database password.
CollectionId	Specifies a unique identifier for the trace. If one is not provided, the default of "0" is used.
Regressflag	Specifies whether the output file is going to be used for testing or not. If set to TRUE, the timestamp information is suppressed so that collected details can be compared across runs.

Creating Database Tables for the Trace Data

Scripts are provided to create tables for trace data.

Run the frmtrc.sql script on the binary trace file to create the database tables, indexes, and views. The frmtrc.sql script calls the frmtrc.tab script (to create tables), the frmtrc.con script (to create constraints), and the frmtrc.vw script (to create views).

The scripts create the following database schema for your trace data:

Table 7–3 Database Schema for Forms Trace Data

Table	Column	Attribute Description
Collections	Collection_Id	Unique ID to identify the collection
	Collection_Name	Name of the trace collection file

Table 7–3 Database Schema for Forms Trace Data

Table	Column	Attribute Description
	Upload_Date	Date on which the data is uploaded
	Trace_Date	Date on which the data is Collected
	User_IP_Address	IP address from which the connection was made
	PID	Process id for the runform process.
User_Actions	Action_Id	Sequence no. to identify the actions within a collection
	Module_Module_Id	Module Id used as a foreign key to Module.
	Collection_ Collection_Id	The collection to which the Action Corresponds. Used as a foreign key to correlate the data
	Action_Context	Context in which the Action took place
	Action_Name	The UserAction
	TimeStamp	Time of Occurrence of the Action
User_Action_ Details	Usr_Action_Id	The action to which the attributes correspond to. Used as a foreign key to correlate the data.
	Collection_ Collection_Id	The collection to which the Action Corresponds. Used as a foreign key to correlate the data
	Attribute_Name	Action Detail Attribute Name
	Attribute_Value	Action Detail Attribute Value
Events	Event_ID	Event Identifier, to identify the sequence of collection events
	Collection_ Collection_Id	The collection to which the Event corresponds to
	Usr_Action_Id	Foreign key to User_Actions
	Event_Event_Id	To identify the associated event. This is useful for duration events.
	Event_Context	Context in which the event took place

Table 7–3 Database Schema for Forms Trace Data

Table	Column	Attribute Description
	Event_Name	Name of the event
	Event_Number	The number assigned to a particular event, for example, 0-3 are errors.
	Timestamp	Timestamp indicating when the event occurred.
	Duration	How long the event took to complete. Only valid for duration events.
Event_Details	Detail_Id	The Detail ID
	Collection_ Collection_ID	The collection to which the Event corresponds to
	Event_Event_Id	The EventID to which the Detail Corresponds to
	Attribute Name	Name of the Attribute
	AttributeValue	Value of the Attribute
Modules	Module_Id	The internal id of the module. Primary key.
	Collection_Collection_ Id	Foreign key to Collection.
	Module_Name	Name of the module, for example, my.fmx.
	Physical_Path	Physical path to the form, for example, d:\temp\my.fmx.
	Attached_Libraries	Names of libraries attached to the form.
	Attached_Menus	Menu attached to the form.

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 Oracle9*i* 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 7–4 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	
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

Table 7–4 List of Traceable Events

Event Number	Definition	Type
46	Timer	point
47	Reserved for future use	
48	Reserved for future use	
49-63	Reserved	
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	
96	Builtin (Start & End)	builtin
97	User Exit (Start & End)	duration
98	SQL (Start & End)	duration
99	MenuCreate (Start & End)	duration
100	PLSQL (Start & End)	duration
101	Execute Query	duration
102-127	Reserved	
128	Client Connect	point
129	Client Handshake	point
130	Heartbeat	point
131	HTTP Reconnect	point
132	Socket (Start & End)	duration

Table 7–4 List of Traceable Events

Event Number	Definition	Type
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	
160-191	Reserved	
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	
1024 an higher	Reserved for internal use	

* 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.

List of Event Details

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

- Errors
- User Action Events
- Forms Services Events

- Detailed Events
- Three-Tier Events
- Miscellaneous

Errors

Lists the error type and the details describing the error.

User Action Events

Table 7–5 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

Forms Services Events

Table 7–6 Forms Services Event Details

Event Name	Details	Number
Form	Form ID, Name, Path, Attached Libraries, Attached Menus	64
Procedure	Procedure Name, FormID	65

Table 7–6 Forms Services Event Details

Event Name	Details	Number
Trigger	TriggerName, FormName, BlockName, ItemName, FormID	66
LOV	LOV name, FormId	67
Editor	FormId , Editor Name	68
Canvas	FormId , Canvas Name	69

Detailed Events

Table 7–7 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

Three-Tier Events

Table 7–8 Three-Tier Event Details

Event Name	Details	Number
Client Connect	Timestamp	128
Client Handshake	Timestamp	129
Heartbeat	Timestamp	130
HTTP Reconnect		131
Socket	FormId, Packets, Bytes	132
HTTP	FormId, Packets, Bytes	133
HTTPS	FormId, Packets, Bytes	134

Table 7–8 Three-Tier Event Details

Event Name	Details	Number
DB Processing	FormId, Statement	135
DB Logon	FormId	136
DB Logoff	FormId	137

Miscellaneous

Table 7–9 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

Servlet Logging Tools

The servlet logging tools available with Oracle9*i*AS Forms Services provides the following:

- A record of all Oracle9*i* Forms sessions, including session start and end times, and the user's IP address and host name (session-level logging)
- Monitoring of Oracle9*i* 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:

- Turning on Logging

- Location of Log Files
- Example Output for Each Level of Servlet Logging

Turning on Logging

Turn on logging by:

- Appending one of the strings in Table 7–10, "Supported logging capabilities" to the serverURL parameter in the URL that starts the form.
- Appending one of the strings in Table 7–10, "Supported logging capabilities" to the serverURL client parameter in the formsweb.cfg file

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 7–10 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.

Specifying Logging in the URL

As an example, to start a performance-level trace, you would start the Oracle9*i* Forms application using a URL as follows:

```
http://yourserver/forms90/f90servlet?serverURL=/forms90/l90servlet/perf
```

Specifying Logging in the formsweb.cfg File

As an example, to start session-level logging for all users, you would change the serverURL entry in the default section of the formsweb.cfg file to the following:

```
serverURL=/forms90/l90servlet/session
```

Specifying Full Diagnostics in the URL Used to Invoke the Forms Servlet

As an example, to start full diagnostics, you would start the Oracle9*i* 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
```

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 Oracle9*i*AS, the full path is:

```
<ORACLE_HOME>/j2ee/ProductGroup2/application-deployments/forms90app/1_default_
island/application.log
```

In Oracle9*i*DS, it is:

```
<ORACLE_HOME>/j2ee/iDS/application-deployments/forms90app/application.log
```

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

(none)

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

/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
```

/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
```

/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
```

/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.
*******************************************
```

Oracle Trace

Oracle Trace is part of Oracle Enterprise Manager (EM). Oracle Trace is the Oracle tool to gather and analyze the performance of the Oracle9*i* Database, Oracle9*i*AS Forms Services, and other products which implement the Oracle Trace API for tracing. To take full advantage of Oracle Trace, you must install Diagnostics Pack, one of the optional packs provided in the Oracle Enterprise Manager product suite. The Diagnostics Pack contains a set of tools to administer the Oracle Trace collections remotely through a GUI interface and to efficiently view the Oracle Trace output.

See the Oracle Enterprise Manager documentation for details.

8

Performance Tuning Considerations

Introduction

This chapter describes the tuning considerations that arise when you use Oracle9*i*AS Forms Services to deploy an application. 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 Oracle9iAS Forms Services Applications

Tuning the connection between Oracle9*i*AS Forms Services and the database server is beyond the scope of this chapter.

Built-in Optimization Features of Forms Services

The Oracle9*i*AS Forms Services and Java client include several optimizations that fit broadly into the following categories:

- 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

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 Oracle9*i* 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 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 Oracle9*i* 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.

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. Oracle9*i*AS 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.

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.

Oracle9*i*AS 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).

Oracle9*i*AS 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), Oracle9*i*AS 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.

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 Oracle9*i* 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 Oracle9*i*AS 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 Oracle9*i*AS Forms Services as a single network message.

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 Oracle9*i* 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 Oracle9*i*AS Forms Services be minimized in order to contain the latency effects of the network.

Tuning Oracle9*i*AS 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 Oracle9iAS Forms Services with Respect to the Data Server
- Minimizing the Application Startup Time
- Reducing the Required Network Bandwidth
- Other Techniques to Improve Performance

Location of the Oracle9*i*AS Forms Services with Respect to the Data Server

The Forms Java client is only responsible to displaying the GUI objects. All of the Oracle9*i* Forms logic runs in Oracle9*i*AS 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 Oracle9*i*AS Forms Services. This allows Oracle9*i* Forms applications to run across slower networks, such as with modems or satellites.

The configuration in Figure 8–1, displays how Oracle9*i*AS Forms Services and the database server are co-located in a Data Center.

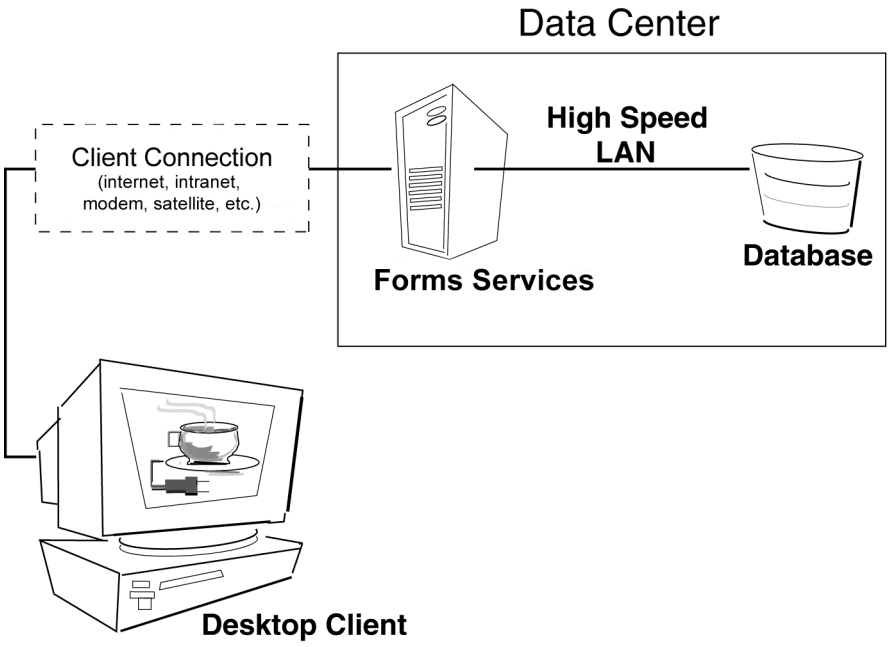


Figure 8–1 Co-Locating the Oracle9iAS Forms Services and Database Server

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 Oracle9*i* 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 Oracle9*i* 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

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.

Oracle9*i*AS Forms Services provides the following pre-configured JAR files to support typical deployment scenarios.

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.

- 90oracle_laf.jar – classes for the Oracle Look-And-Feel
- 90generic_laf.jar – classes for the generic (standard) Look-And-Feel
- 90resources.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".

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
```

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
```

Using Caching

Both of the supported JVMs for Oracle9*i*AS 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.

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.

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.

A

JInitiator

Oracle JInitiator

This section describes the benefits of using Oracle JInitiator as a Web browser plug-in. Oracle JInitiator enables users to run Oracle9*i* 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.

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 Oracle9*i* 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 Oracle9*i* 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).

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 Oracle9*i*AS 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.

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 Oracle9*i* Forms 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 Oracle9*i* Forms applications as trusted applets in the Oracle JInitiator environment.

Supported Configurations

Oracle JInitiator supports the following configurations:

Windows 98, NT, 2000, XP:

- Navigator 4.7.3
- Navigator 4.7.8
- Internet Explorer 5.x

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)

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.

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.

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.

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 Chapter 3, Default basejini.htm File.

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.

Making Oracle JInitiator available for download

To make Oracle JInitiator available for download:

1. Copy jinit13*x*.EXE to your Web server.

   You must copy jinit13*x*.EXE to the location that was specified within the base HTML file.
2. Copy JINIT_DOWNLOAD.HTM to your Web server.

You must copy JINIT_DOWNLOAD.HTM to the location that was specified within the base HTML file.

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.

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.

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.

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.

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.

Modifying the base HTML file

When you run an Oracle9*i* Forms application with the help of JInitiator, JInitiator reads parameter values from the formsweb.cfg file and passes these values into the baseHTML file. If you want to create a static baseHTML file so that the same values are read all the time, you need to manually place them in the baseHTML file.

For an example of the Oracle JInitiator markup for both Microsoft Internet Explorer and Netscape Navigator, see Chapter 3 Default basejini.htm File. Adding these tags to your baseHTML file will enable your applications to run within both Netscape and Microsoft browsers.

Index

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X