Deploying Applications on the Web

Release 1.6.1

January, 1998
Part No. A57514-03
## Contents

1 Understanding the Developer/2000 Server

1.1 Introduction to the Developer/2000 Server ............................................................... 1-1

1.1.1 The best of the Web and client/server ................................................................. 1-1

1.1.2 Three-tiered architecture ...................................................................................... 1-2

1.2 Benefits of the Developer/2000 Server ................................................................. 1-3

1.3 Forms Web architecture .......................................................................................... 1-4

1.3.1 About the Forms Client ....................................................................................... 1-5

1.3.2 About the Forms Server ....................................................................................... 1-5

1.3.2.1 Client/server architecture vs. Web architecture ........................................... 1-6

1.3.2.2 Call and response: Forms Client and Forms Server .................................. 1-7

1.3.3 Running Forms applications on the Web ........................................................... 1-8

1.3.4 Security and encryption ....................................................................................... 1-9

1.4 Reports Web architecture ....................................................................................... 1-10

1.4.1 Static Reporting with the Web Wizard ............................................................... 1-11

1.4.2 Scheduling Reports with the Reports Server .................................................... 1-11

1.4.3 Dynamic Reporting with the Web Cartridge or CGI ........................................ 1-11

1.4.3.1 About the Reports Server ............................................................................ 1-12

1.4.3.2 About the Reports Web Cartridge ............................................................... 1-12

1.4.3.3 About the Reports Web CGI ........................................................................ 1-12

1.5 Graphics Web architecture ..................................................................................... 1-14

1.5.1 About the Graphics Client and Graphics Server .............................................. 1-14

1.5.2 About the Graphics Client .................................................................................. 1-15

1.5.3 About the Web server with Oracle Web Request Broker module .................... 1-15

1.5.4 About the Graphics Server .................................................................................. 1-15

1.5.5 Call and response: Graphics Client and Graphics Server ................................ 1-15

1.5.6 About display instances and session IDs ......................................................... 1-16

1.5.6.1 Accessing Graphics displays via the web interface toolbar ......................... 1-16

1.5.6.2 Accessing Graphics displays via parameterized URLs ................................ 1-17

1.5.7 About using Graphics parameters in URLs ....................................................... 1-18
2 Setting Up the Developer/2000 Server

2.1 Configuring the Forms Server ................................................................. 2-1
  2.1.1 Step 1: Generating and deploying .FMX files ................................. 2-2
  2.1.2 Step 2: Starting and stopping the Forms Server Listener ............... 2-2
  2.1.2.1 Starting a Forms Server Listener process ................................. 2-3
  2.1.2.2 Checking the status of the Forms Server Listener ...................... 2-3
  2.1.2.3 Stopping a Forms Server Listener process .................................. 2-4
  2.1.3 Step 3: Making applications available to end users ....................... 2-4
  2.1.3.1 Step 3a: Creating virtual directories on your Web server .......... 2-5
  2.1.3.2 Step 3b: Choosing an HTML file construction method ............... 2-6
  2.1.3.3 Creating an application cartridge ............................................. 2-6
  2.1.3.4 Creating a base HTML file for the application cartridge .......... 2-10
  2.1.3.5 Creating a static (non-cartridge) HTML file .............................. 2-10
  2.1.3.6 Step 3c: Providing URLs of your Web-enabled applications ....... 2-11
  2.1.4 Step 4: Configuring the Forms Client ........................................... 2-12
  2.1.4.1 About JAR files .......................................................................... 2-13
  2.1.4.2 About the generic Oracle JAR file ............................................. 2-13
  2.1.4.3 Creating custom JAR files ....................................................... 2-13
  2.1.4.4 Referencing JAR files .............................................................. 2-14
  2.1.4.5 Example of adding the ARCHIVES tag to an application’s static HTML file: 2-15
  2.1.5 Setting up load balancing .............................................................. 2-15
  2.1.5.1 Configuring your Web cartridge for load balancing ................. 2-19
  2.1.5.2 Installing the Metrics Server and Clients .................................. 2-21
  2.1.5.3 Modifying your cartridge base HTML file ............................... 2-22
  2.1.5.4 Starting the Metrics Server ...................................................... 2-22
  2.1.5.5 Starting the Metrics Clients ..................................................... 2-23
  2.2 Troubleshooting your Forms configuration ........................................ 2-25
  2.3 Configuring the Reports Server .......................................................... 2-28
  2.3.1 Step 1: Setting up the Reports Server ........................................... 2-28
  2.3.1.1 Setting up for Windows 95 ...................................................... 2-28
  2.3.1.2 Setting up for Windows NT as a service ................................. 2-29
  2.3.1.3 Setting up for Windows NT as a non-service ......................... 2-31
  2.3.1.4 Setting up for UNIX ............................................................... 2-32
  2.3.2 Step 2: Setting up the Reports Web Cartridge or Web CGI .......... 2-33
  2.3.2.1 Setting up the Reports Web Cartridge .................................... 2-33
  2.3.2.2 Setting Up the Reports Web CGI for Windows 95/NT ............... 2-36
  2.3.2.3 Setting Up the Reports Web CGI for UNIX ............................. 2-36
  2.3.2.4 Cartridge Parameters and Environment Variables ................... 2-37
  2.3.2.5 ORACLE_HOME ................................................................. 2-38
  2.3.2.6 LD_LIBRARY_PATH ............................................................. 2-38
2.3.2.7 REPORTS25_WEBLOC .......................................................... 2-38
2.3.2.8 REPORTS25_WEBLOC_TRANSLATED .................................................. 2-39
2.3.2.9 REPORTS25_OWSHELP ............................................................ 2-39
2.3.2.10 REPORTS25_OWSMAP ............................................................. 2-40
2.3.2.11 REPORTS25_OWSDIAGBODYTAGS .............................................. 2-40
2.3.2.12 REPORTS25_OWSDIAGHEADTAGS .............................................. 2-40
2.3.2.13 REPORTS25_OWSNODIAG .......................................................... 2-41
2.3.2.14 REPORTS25_OWSNODIAG .......................................................... 2-41
2.3.2.15 REPORTS25_CGIIHP ................................................................. 2-41
2.3.2.16 REPORTS25_CGIMAP ................................................................. 2-41
2.3.2.17 REPORTS25_CGIDIAGBODYTAGS .............................................. 2-41
2.3.2.18 REPORTS25_CGIDIAGHEADTAGS .............................................. 2-42
2.3.2.19 REPORTS25_CGINODIAG .......................................................... 2-42
2.3.2.20 REPORTS25_CGIPATHONLYURL ................................................ 2-42
2.3.2.21 USER_NLS_LANG or NLS_LANG .................................................... 2-42
2.3.3 Step 3: Specifying URL run requests .................................................... 2-42
2.3.3.1 A non-URL-encoded (non-standard) request ..................................... 2-43
2.3.3.2 A URL-encoded standard request .................................................... 2-43
2.3.3.3 A non-URL-encoded (non-standard) form-style request ..................... 2-43
2.3.3.4 A URL-encoded standard form-style request ..................................... 2-44
2.3.3.5 An HTML form: .............................................................................. 2-44
2.3.4 Step 4 (Optional): Using the URL mapping feature ................................ 2-45
2.3.5 Step 5: Deploying reports on the Web .................................................. 2-46
2.4 Troubleshooting your Reports Server configuration ..................................... 2-47
2.5 Configuring the Graphics Server ............................................................. 2-49
2.5.1 Creating virtual directories on your Web server ................................... 2-49
2.5.2 Creating the Graphics Server cartridge ............................................. 2-49
2.5.3 Making the Graphics web cartridge available to end users ..................... 2-54

3 Guidelines for Designing Web Applications

3.1 General guidelines for designing Web applications ................................. 3-1
3.1.1 Using RUN_PRODUCT to integrate Developer/2000 applications .......... 3-2
3.1.1.1 Running Reports Reports from a Web-enabled Forms application .... 3-2
3.1.1.2 Running Graphics from a Web-enabled Forms application ............. 3-3
3.1.2 Creating your own template HTML files ............................................ 3-3
3.1.3 Creating an HTML application menu ............................................... 3-3
3.2 Guidelines for designing Forms Web applications .................................. 3-5
3.2.1 Using Designer/2000 with the Developer/2000 Server ....................... 3-5
3.2.2 Reducing network traffic ................................................................. 3-5
3.2.3 Creating a generic GUI for Web-enabled Forms applications ............. 3-6
3.2.4 Selecting fonts ................................................................. 3-6
3.2.5 Using the SHOW_DOCUMENT built-in .............................................. 3-6
3.2.6 Working with images ................................................................. 3-7
3.2.7 Feature restrictions for forms on the Web ............................................. 3-7
3.3 Guidelines for designing Reports Web applications .............................. 3-8
  3.3.1 About Web reports ........................................................................ 3-8
  3.3.2 Adding Web functionality .............................................................. 3-9
  3.3.2.1 Hyperlinks ................................................................................... 3-9
  3.3.2.2 Bookmark .................................................................................. 3-13
  3.3.2.3 Action ....................................................................................... 3-14
  3.3.2.4 Report Escapes ........................................................................ 3-14
  3.3.3 Selecting the output format ......................................................... 3-15
  3.3.3.1 About HTML output ................................................................. 3-16
  3.3.3.2 About PDF format ..................................................................... 3-19
  3.3.4 Helpful tips .................................................................................. 3-20
  3.4 Guidelines for designing Graphics web applications ............................ 3-20
    3.4.1 Using custom hyperlinks in Graphics displays ............................... 3-21

A Working with Template HTML Files
  A.1 About Template HTML Files ........................................................... A-1
  A.2 Working with the Cartridge HTML File Template ................................ A-2
    A.2.1 Snapshot: the cartridge HTML file template .................................. A-3
    A.2.2 Modifying the cartridge HTML file template .................................. A-4
    A.2.3 Examples of customized cartridge HTML files ............................... A-6
    A.2.3.1 Example 1 .................................................................................. A-6
    A.2.3.2 Example 2 .................................................................................. A-7
  A.3 Working with the Static (Non-Cartridge) HTML File Template ............... A-8
    A.3.1 Snapshot: the static (non-cartridge) HTML file template ................ A-8
    A.3.2 Customizing the static (non-cartridge) HTML file template ............ A-9
    A.3.3 Examples of customized static (non-cartridge) HTML files ............... A-10
    A.3.3.1 Example 1 .................................................................................. A-10
    A.3.3.2 Example 2 .................................................................................. A-10

B Including Application-Specific Settings
  B.1 About Application Classes ............................................................. B-1
  B.2 Referencing an application class ....................................................... B-2
  B.3 Setting an icon directory path for an application class ......................... B-3
  B.4 Creating font-mapping settings for an application class ....................... B-4
C Release-Specific Information

C.1 Creating an Application Cartridge for Your Forms Server ............................................. C-1
C.1.1 Creating and registering the cartridge, and creating its virtual directory ............. C-1
C.1.2 Creating a base HTML file for the cartridge .......................................................... C-7
C.2 Setting Up the Reports Web Cartridge ........................................................................ C-8
C.2.1 Setting Up for Windows 95/NT .............................................................................. C-8
C.2.2 Setting Up for UNIX .............................................................................................. C-10
C.3 Creating the Graphics Server Cartridge Entry in the Web Request Broker ............. C-13
C.3.1 Setting cartridge parameters and values .............................................................. C-13

Index ...................................................................................................................... Index-1
Send Us Your Comments

Developer/2000: Deploying Applications on the Web
Part No. A57514-03

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

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

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number.

You can send comments to us in the following ways:

■ Email: d2kdoc@us.oracle.com
■ Fax: 415/506-7419 (Attn: Developer/2000 Documentation)
■ Postal service:
  Oracle Corporation
  Developer/2000 Documentation
  500 Oracle Parkway, Box 659209
  Redwood Shores, CA  94065
  U.S.A.

If you would like a reply, please give your name, address, and telephone number below.

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________
Preface

Developer/2000 for the Web enables you to quickly and easily deploy new or existing Developer/2000 applications on the World Wide Web. This manual describes the benefits, setup procedures, and design guidelines for developing and deploying Web applications.

Structure

This manual contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td>Introduces Developer/2000 for the Web.</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Explains how to set up Developer/2000 for the Web so you can deploy new or existing applications on the Web.</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Provides guidelines for designing applications for the Web.</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Provides the following for the cartridge and static (non-cartridge) template HTML files provided by Oracle:</td>
</tr>
<tr>
<td></td>
<td>■ the directory path and filename of the template file</td>
</tr>
<tr>
<td></td>
<td>■ a snapshot of the template file</td>
</tr>
<tr>
<td></td>
<td>■ instructions for modifying the template file’s parameters and values</td>
</tr>
<tr>
<td></td>
<td>■ examples of customized template files</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Describes how to use application classes to customize Web-enabled Forms applications.</td>
</tr>
<tr>
<td>Appendix C</td>
<td>Provides instructions if you are using Oracle WebServer 2.1 rather than Oracle Web Application Server 3.0.</td>
</tr>
<tr>
<td>Glossary</td>
<td>Defines terms and acronyms used throughout the manual.</td>
</tr>
</tbody>
</table>
Related Documents

As an application developer building and deploying Developer/2000 applications on the Web, you may need to refer to some or all of the documents listed below:

- *Forms 4.5 Developer’s Guide*, Part No. A32505
- *Reports 2.5 Building Reports Manual*, Part No. A32488-1
- *Graphics 2.5 Developer’s Guide*, Part No. A32482-1
- *Oracle7 Server Concepts Manual*, Part No. 6693-70-1292
- *Understanding SQL*Net*, Part No. A48385-1

Conventions

The following typographic conventions are used in this manual:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface text indicates a control (such as a button or a text field) you must use to complete a task.</td>
</tr>
<tr>
<td><em>italics</em></td>
<td>Italicized text indicates new terms or acronyms defined in the text, the Glossary, or both. Italicized text also signifies terms in a list.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace text is used for code examples and user-entered text.</td>
</tr>
<tr>
<td>sans serif</td>
<td>Sans serif text indicates the correct syntax of a command, filename, parameter, and so on.</td>
</tr>
<tr>
<td><code>&lt; &gt;</code></td>
<td>Angle brackets denote user-supplied values.</td>
</tr>
</tbody>
</table>
This chapter covers these topics:

- Introduction to the Developer/2000 Server
- Benefits of the Developer/2000 Server
- Forms Web architecture
- Reports Web architecture
- Graphics Web architecture

1.1 Introduction to the Developer/2000 Server

The Developer/2000 Server is a new generation of Oracle development tools that enable you to deploy new and existing applications on the World Wide Web, either on an internal company intranet, or on the Internet. The Developer/2000 Server 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, dynamic applications.

1.1.1 The best of the Web and client/server

The Developer/2000 Server provides solutions that enable you to take advantage of all the benefits of the Web, while maintaining the strengths of client/server computing. The Web greatly reduces the costs of administering and maintaining applications, while allowing for a thin, low-cost client. At the same time, it allows you to leverage your existing client/server applications, which may be mission-critical applications that support—and often drive—your business practices. These applications still must be scalable to large numbers of users and open to all of your environments.
1.1.2 Three-tiered architecture

In most client/server implementations today, running applications is a highly client-intensive process. Though data is extracted from a remote database server, applications run on client machines, which often have limited processing power and memory capacity.

The Developer/2000 Server supports a three-tiered architecture that delivers the benefits of both client/server and the Web in a single application. In a Web implementation, application logic and processing are focused on a middle tier of application servers instead of on desktop client machines.

<table>
<thead>
<tr>
<th>Tier</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>front-end</td>
<td>any number of client desktop machines</td>
</tr>
<tr>
<td>middle</td>
<td>one or more application servers</td>
</tr>
<tr>
<td>back-end</td>
<td>one or more database servers</td>
</tr>
</tbody>
</table>

*Figure 1–1 Three-tiered architecture*
1.2 Benefits of the Developer/2000 Server

The Developer/2000 Server benefits groups across the spectrum of the application life-cycle, including managers, developers, and end users. Each group benefits by gains in productivity, reduced resource requirements, and lower costs:

- **Low-cost deployment and maintenance.** Web applications are deployed and maintained on centralized application servers, from which they download to end users’ **Web browsers** at runtime. To roll out a Web application, simply give end users the application’s **URL**. This distribution method reduces the time, cost, and complexity of deploying applications to a large or geographically-dispersed end user base, all without installing application software on their desktop machines.

- **Client/server migration.** The **Developer/2000 Server** extends and leverages the traditional strength of Oracle’s client/server development suite. Developer/2000 supports all **GUI** widgets available in a client/server application, so you can Web-deploy any existing client/server application without changing the form definition (.FMB file)\(^1\). You can run the same Form executable (generated .FMX file) in client/server mode or on the Web. Programmers experienced with Developer/2000 do not need to learn Java, JavaScript, or any other new language.

- **“Thin client” architecture.** The only client-side requirement is a Java-enabled\(^2\) Web browser. Any end user with a Java-enabled browser can run Web-enabled applications. This greatly reduces processor and memory requirements for end users’ desktop machines.

- **Support for various business objectives.** With the Developer/2000 Server, you can develop internal, transaction-intensive applications (deployed on a company intranet), and electronic commerce applications (deployed on the Internet). End users accustomed to client/server applications will enjoy similar performance with the Developer/2000 Server.

- **NC support.** The **Developer/2000 Server** is designed in accordance with Oracle’s **Network Computing Architecture (NCA)**. Developer/2000 applications run as NCA-compliant Web cartridges.

---

\(^1\) See Section 3.2.7, “Feature restrictions for forms on the Web”.

\(^2\) Java 1.1.X (see your Developer/2000 release notes)
- **Platform independence.** The Developer/2000 Server leverages Java—the emerging standard platform-independent language—to map GUI widgets to their native counterparts on other platforms. The look and feel of widgets will differ only slightly between end user platforms (Windows, Macintosh, Motif, and so on). For more information, see Section 3.2.3, “Creating a generic GUI for Web-enabled Forms applications”.

- **Web publishing.** Reports provides you with a number of features that enable Web publishing of your reports.

- **Web Wizard.** You can quickly Web-enable your existing reports using the Web Wizard in Reports. The Web Wizard enables you to specify Web features for the report and immediately generate output in Hypertext Markup Language (HTML) or Portable Display Format (PDF).

- **Drill-down reporting.** You can associate URLs with objects in a report. By associating a URL with an object in a report, you enable end users to easily navigate to other Web pages or launch other reports that provide more detailed or related data.

- **Dynamic reporting.** To dynamically run reports from a Web browser, you can use the Reports Multi-tier Server (hereafter called the Reports Server) in conjunction with the Reports Web Cartridge or Web CGI. When a user clicks on a URL that calls a report, the Reports Cartridge or Web CGI transfers the request to the Reports Server. The Reports Server queues and runs the report, generating the specified output. The Web server can then serve the report output back to the user in their Web browser.

- **Monitoring the report queue.** Using the Reports Queue Manager, you can administer the Reports Server queue, launch reports, and schedule reports to run automatically at specific times or intervals.

### 1.3 Forms Web architecture

To run new or existing Forms applications on the Web, you should install and operate Developer/2000 on the middle tier of a distributed three-tiered architecture (see Figure 1–1, “Three-tiered architecture”).

The Forms Web component of Developer/2000 consists of the *Forms Client* and the *Forms Server*. 
1.3.1 About the Forms Client

The Forms Client is a Java applet—downloaded at runtime from an application server to an end user’s Web browser—that displays the form’s user interface and manages interaction between end users and the Forms Server. The Forms Client receives “bundles” of interface commands from the Forms Server and translates them (in sets) into interface objects for the end user. Some interface events handled by the Forms Server Runtime Engine in a client/server implementation (such as typing characters in a text field, or moving around a dialog) occur only on the Forms Client in the Web implementation, with no interaction with the Forms Server runtime engine.

The Forms Client is:

- **Generic.** You are not required to deploy a separate Java applet for each application you wish to deploy on the Web.
- **Dynamic.** The Forms Client dynamically reacts to the current form at runtime, requesting and displaying only the information and user interface elements necessary to represent the current state of the application at any given time.
- **Feature-rich.** The Forms Client supports all user interface widgets and tools available in a client/server implementation. Due to Java object standards, the look and feel of some Forms widgets may vary slightly when deployed on the Web. For more information, see Section 3.2.7, “Feature restrictions for forms on the Web”.
- **Thin.** At startup, only those class files necessary to render the initial state of an application are downloaded to the end user’s machine. Additional class files are downloaded dynamically (as needed) to support additional user interface functionality.

1.3.2 About the Forms Server

The Forms Server consists of two components:

- **Listener.** The Forms Server Listener initiates the Forms Server runtime session and establishes a connection between the Forms Client and the Forms Server Runtime Engine.

- **Runtime Engine.** The Forms Server runtime engine is a modified version of the Forms 4.5 Runtime Engine, with user interface functionality redirected to the Forms Client. It handles all form functionality except UI interaction, including trigger and commit processing, record management, and general database interaction.
1.3.2.1 Client/server architecture vs. Web architecture

There are two main differences between deploying Forms applications in a client/server implementation and a Web implementation:

- **Client/server.** The Forms Server runtime engine (and all application logic) are installed on end users’ desktop machines. Although your application can include database-server-side triggers and logic, typically all user interface and trigger processing occurs on client machines.

![Diagram of Client-Server architecture](image)

*Figure 1–2 Developer/2000: client/server architecture*
The Forms Server runtime engine (and all application logic) are installed on application servers, not on client machines. All trigger processing occurs on database and application servers, while user interface processing occurs on the Forms Client.

**Figure 1–3 Developer/2000: Web architecture**

### 1.3.2.2 Call and response: Forms Client and Forms Server

Once a direct network connection is established between the Forms Client and Forms Server, the two components communicate through a series of requests and responses—via compressed messages passed over a network.

Requests from the Forms Client are events (such as “click button” or “display LOV”). Responses from the Forms Server are a series of changes to the user interface (such as value changes, and adding and removing components), all of which the Forms Client turns into display objects.

For example, the Forms Client might receive a response from the Forms Server similar to “create a green text item on canvas CAN_12.” The Forms Client translates the response into actual interface objects, in this case, the colorful text item.
The Forms Client contacts the Forms Server when users perform:

- High-level operations (such as accepting or canceling a dialog).
- Operations that involve validation processing and cause default and user-defined triggers to fire (such as checking a checkbox or navigating between fields).

### 1.3.3 Running Forms applications on the Web

**Figure 1–4 Running Forms applications on the Web: process flow**
To start and run a Forms application on the Web, end users use a Java-enabled Web browser to access a URL. The following sequence occurs automatically:

1. The URL corresponds either to a static (non-cartridge) HTML page, or to an application cartridge, residing on the application server.
2. An HTML page, and then the Forms Client applet are downloaded from the application server to the user’s browser.
3. The Forms Client sends a request to the Forms Server Listener (which resides on a specific port of the machine from which the Forms Client was downloaded).
4. The Listener contacts the Forms Server runtime engine and connects to a Forms Server runtime process (either by starting a new process, or by connecting to an existing process). If included in the HTML page, Forms command-line parameters (such as form name, user ID and password, database SID, menu name, and so on) and any user-defined Forms parameters are passed to the process by the Listener.
5. The Listener establishes a direct socket connection with the Runtime Engine, and sends the socket information to the Forms Client. The Forms Client then establishes a direct socket connection with the Runtime Engine. The Forms Client and Runtime Engine then communicate directly, freeing the Listener to accept startup requests from other end users. The Forms Client displays the application’s user interface in an applet window outside the main window of the end user’s Web browser.
6. As in a client/server implementation, the Runtime Engine communicates directly with the database through SQL*Net (or another driver, for non-Oracle data sources).

1.3.4 Security and encryption

Data passed between the database, the Forms Server, and the Forms Client is automatically encrypted before—and decrypted after—transmission by the following protocols:

- **RSA RC4 40-bit encryption** (for transmissions between the Forms Client and the Forms Server)
- **SQL*Net SNS/ANO** (for transmissions between the Forms Server and the database server)

**Note:** Encryption is provided by default, but can be disabled.
1.4 Reports Web architecture

The Reports Web architecture provides a flexible way to publish information on the Web. It enables you to tailor your Web reporting to the needs of your data.

One method of Web publishing is to generate static HTML or PDF output, place the files on a server, and link them to a Web page to make them accessible to users. This method is very useful when you run reports at regular intervals and keep previous versions of the output for sometime. For example, a sales report for the third quarter of the current fiscal year would be generated after the close of the quarter and the output would remain available indefinitely. A highlight report for your department might be published once per week and reports from previous weeks remain available indefinitely.

For other data (for example, inventory), though, it may be imperative that you see the very latest information available. Dynamic Web reporting generates report output as you navigate to it in your Web browser. When you click a URL, a report is executed and the output is served back to you in your Web browser. This method ensures that you are looking at the very latest information.

Reports supports all of these scenarios with its Web Wizard, Report Editor, and Reports Server. The Web Wizard enables you to add basic Web properties (for example, bookmarks) to an existing report and immediately generate HTML or PDF output. If the output only needs to be generated once, the Web Wizard may be all that you need.

If the output needs to be refreshed at regular intervals (for example, once per week), you can automate the report’s execution. After using the Web Wizard to add Web properties to the report, you can use the Reports Server to schedule the report to run at regular intervals, regenerating the HTML or PDF output each time.

For dynamic Web reporting, you can use the Reports Server in conjunction with the Reports Web Cartridge or Web CGI. When you click on a link, the URL is processed by the Reports Web Cartridge or Web CGI, and a job request is sent to the Reports Server. The Reports Server sends the job to a runtime engine for execution. When the report output is ready, it can be served back to the Web browser by the Web server.
1.4.1 Static Reporting with the Web Wizard

To enable you to quickly prepare existing the Reports Server, Reports provides a Web Wizard (accessible from the toolbar and Tools menu). The Web Wizard contains a series of pages that let you specify:

- Header and footer HTML files to be added at the beginning and end of the report output.
- Bookmarks to be added to break columns to create an outline for the report.
- The type of output to be generated (HTML, HTML style sheets, or PDF).
- Whether to generate output immediately.

Note: In addition to the Web Wizard, you can also use the Report Editor to add Web functionality to a report. You should use the Web Wizard to add the basic Web functionality (for example, bookmarks) and the Report Editor to add more advanced Web functionality (for example, customized Report Escapes).

1.4.2 Scheduling Reports with the Reports Server

To enable you to schedule reports to run at regular intervals, the Reports Queue Manager and the R30CLI command line provide scheduling options. In the Reports Queue Manager, you can specify scheduling options whenever you launch a new job from the Job menu. On the R25CLI command line, you can use the SCHEDULE argument to specify when to run the report.

1.4.3 Dynamic Reporting with the Web Cartridge or CGI

To enable end users to generate reports dynamically from a Web browser, the Developer/2000 Server provides a Reports Server suite that includes the following components:

- The Reports Server (R25MTS), which enables you to run reports on a remote server.
- The Reports Web Cartridge (R25OWS), which sends requests from the Web server to the Reports Server if you are using the Oracle Web Application Server 3.0.
- The Reports Web Common Gateway Interface (R25CGI), which sends requests from the Web server to the Reports Server using standard CGI.
1.4.3.1 About the Reports Server
The Reports Server enables you to run reports on a remote, application server. When used in conjunction with the Reports Web Cartridge or Web CGI, it also enables you to run reports from a Web browser using standard URL syntax. The server can be installed on Windows NT, Windows 95, or UNIX. It handles client requests to run reports by entering all requests into a job queue. When one of the server’s runtime engines becomes available, the next job in the queue is dispatched to run. As the number of jobs in the queue increases, the server can start more runtime engines until it reaches the maximum limit specified when the server process was started. Similarly, idle engines are shut down after having been idle for longer than a specified period of time. You can monitor and administer the job queue on Windows using the Reports Queue Manager (R25RQM32.EXE). On UNIX, you can use the Reports Queue Viewer (r25rqv)

1.4.3.2 About the Reports Web Cartridge
The Reports Web Cartridge is a library (on Windows NT, a dynamically linked library or .dll file; on UNIX, a shared library or .so file). It performs the same functions as the Reports Web CGI, but it has the advantage of native integration with the Oracle Web Application Server 3.0. Once installed, it is always up and ready. The CGI executable has to be started up every time it is invoked. The cartridge typically resides in the $ORACLE_HOME/bin directory and is installed using the Web Request Broker of an active Oracle Web Application Server Listener.

1.4.3.3 About the Reports Web CGI
The Reports Web CGI provides a standard CGI connection between a Web server and the Reports Server. This connection enables a Web client to run reports using standard URL syntax and see the output on the client’s Web browser. R25CGI is designed to run under WWW CGI v1.1.
To start a report on the Web, end users point to a URL from their Web browsers. The following sequence then occurs:

1. The Web browser passes the URL to the Web server, and the Web server sends the request to the Reports Web Cartridge or CGI.

2. The Reports Web Cartridge or CGI parses the request and converts it to a command line that can be executed by the Reports Server. Then it submits the command line to the specified Reports Server (synchronously).

3. The Reports Server queues the job request from the Web Cartridge or CGI and, when one of its runtime engines becomes available, it runs the report.

Figure 1–5 Developer/2000: Reports Web architecture
4. The Reports Web Cartridge or CGI retrieves the name of the report output from the server and creates HTTP redirection to the output file.

5. The Web server executes HTTP redirection, passing the new URL back to the Web browser, and the report is displayed on the client.

1.5 Graphics Web architecture

The three-tiered architecture of the Developer/2000 Server enables you to enhance your Web publications with interactive, data-driven graphic displays that can be accessed from a remote application server by any client. You can leverage this capability to embed powerful graphical applications directly in HTML pages, or in Forms applications on the Web.

1.5.1 About the Graphics Client and Graphics Server

The Graphics web architecture consists of three components, as illustrated in Figure 1–6, “Graphics Web architecture”:

- Graphics Client
- Oracle Web Application Server 3.0 with Web Request Broker component
- Graphics Server

![Figure 1–6 Graphics Web architecture](image)
1.5.2 About the Graphics Client

The Graphics Client displays an interface that resembles the Graphics Runtime interface and allows you to run your existing Graphics applications without modification on the Web. Any HTML Web browser running on any platform can function as a Graphics Client.

1.5.3 About the Web server with Oracle Web Request Broker module

The Web Request Broker is a separate module provided with Oracle Web Application Server 3.0 and supported by Netscape, Microsoft Web Server and others. The Web Request Broker provides a framework for, and manages client connections to, cartridges such as the Graphics Server.

1.5.4 About the Graphics Server

The Graphics Server is a variation of the Graphics Runtime engine that runs on the Web server and functions as an application server. When invoked by requests or events submitted by the client and relayed through the Web Request Broker, the Graphics Server processes the event, and returns an HTML page containing a reference to an updated display image. The server communicates with the database through SQL*Net (as it does in the traditional client/server implementation).

1.5.5 Call and response: Graphics Client and Graphics Server

The Graphics Client and Graphics Server communicate via an HTTP connection through a series of requests and responses relayed through the Web Request Broker. Requests are initiated on the client-side by direct URL submission, mouse-clicks in the display region, or HTML-form submit requests. All requests are passed as one of the following:

- A simple URL.
- A parameterized URL containing special keyword arguments similar to those used in Graphics Batch (for example, userids or values for user-defined parameters).

Responses are returned as one of the following:

- An HTML form functioning as a Graphics web interface.
- An HTML form containing an image of the requested display.
1.5.6 About display instances and session IDs

It is possible for multiple instances of a Graphics display to be open simultaneously if:

- A single user uses more than one browser to open multiple instances of a display at the same time.
- Different users open the same display at the same time.

When multiple instances of a display are open, each instance of the display is unique and independent. That is, when a user performs an action on one instance of a display, other instances of the display are not affected.

The Graphics Server uses session IDs to identify instances of open displays. When the server receives a request for a display, it generates a unique session ID representing the display instance, and embeds the ID in its initial response to the client. The client returns the session ID as a parameter with any event it submits to the server, thus identifying the instance of the display that is being referenced. Note that the generation and submission of session IDs is an automatic process and requires no attention from the end user or developer.

1.5.6.1 Accessing Graphics displays via the web interface toolbar

The easiest way to access a Graphics display is through the Graphics web interface toolbar. To use this method, the user points the web browser to the simple URL that invokes the toolbar. For example:

```
http://my_server/ogweb
```

The following sequence then occurs:

1. The client passes the URL to the Web server.
2. The server’s Web Request Broker forwards the request to the Graphics Server.
3. The server returns an HTML-form that functions as the Graphics web interface toolbar.
4. On the toolbar, the user types a username and password to connect to the database, selects a display from the list of available displays, and clicks Open. This causes the browser to submit a request in the form of a parameterized URL similar to the following:

```
http://my_server/ogweb?openfile=my_display.ogd&userid=scott/tiger@og73&my_parameter=my_value
```
5. The Web Request Broker forwards this request to the Graphics Server.

6. The server returns an HTML page containing the requested image.

7. Each time the user executes an event—for example, by clicking the mouse in a display region—the client generates another request. Events are relayed via automatically-generated parameterized URLs containing keywords describing the details of the event (for example, mouse X and Y positions).

1.5.6.2 Accessing Graphics displays via parameterized URLs

Users familiar with the standard keywords used to invoke Graphics Batch from a command line can construct URLs containing the necessary parameters to access Graphic displays directly from the server, rather than through the Graphics web toolbar. This method of accessing a display enables you to:

- Embed Graphics displays in custom-HTML pages.
- Execute PL/SQL code contained in displays.
- Supply additional information (such as userids, or values for user-defined parameters) in display requests.
- Turn the Graphics web interface toolbar on or off.

To accomplish any of the above, the user submits a parameterized URL. For example:

```
http://www.my_company.com/ogweb?openfile=my_display.ogd&userid=scott/tiger@og73&oracle_interpret="BEGIN; MY-PROC(my_argument); END;"
```

The following sequence then occurs:

1. The client passes the URL to the Web server.
2. The server’s Web Request Broker forwards the request to the Graphics Server.
3. The server performs the operations specified by any additional keywords, and returns an HTML page containing an image of the requested display.
4. Each time the user executes an event—for example, by clicking the mouse in a display region—the client generates another request. Events are relayed via additional, automatically-generated, parameterized URLs containing keywords describing the details of the event (for example, mouse X and Y positions).
For more information about using Graphics command line options in parameterized URLs, see Section 1.5.7, “About using Graphics parameters in URLs”.

### 1.5.7 About using Graphics parameters in URLs

There are a number of Graphics keywords that can appear as “name-value” pairs in URLs used to invoke the Graphics Server. For example, a user might submit the following URL:

```
http://www.myserver.com/ogweb?openfile=mydisplay.ogd&userid=scott/tiger@og73 &showtoolbar=yes
```

This request employs the `userid`, `openfile`, and `showtoolbar` keywords to connect to a database, open a display, and turn on the Graphics web interface toolbar.

The following table defines the valid keywords and specifies the range of possible values for each:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Use</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>openfile</td>
<td>Specifies the name of the Graphics display file (.OGD) to open.</td>
<td>The name of any valid Graphics display file. A suffix of .OGD or .OGR is assumed if not specified. For example: openfile=my_display.ogd</td>
</tr>
<tr>
<td>userid</td>
<td>Specifies the full connect string to log on to the desired database.</td>
<td><code>userid/password@dbname</code> For example: userid=scott/tiger@og73</td>
</tr>
<tr>
<td>showtoolbar</td>
<td>Turns Graphics web toolbar on or off.</td>
<td>`{yes</td>
</tr>
<tr>
<td>sessionid</td>
<td>The unique identifier for the instance of a Graphics display being viewed via the Graphics Server. For more information about session IDs, see Section 1.5.6, “About display instances and session IDs”.</td>
<td>Any alpha-numeric session ID number generated by the Graphics Server. For example: sessionid=000001.091032</td>
</tr>
</tbody>
</table>
**Note:** All URL submissions to the Graphics Server must use the standard URL format of changing spaces to +, and encoding special characters with %xx hexadecimal encoding. For more information about encoding URLs, see the W3C (World Wide Web Consortium) document “Names and Addresses, URIs, URLs, URNs, URCs” at http://www.w3.org/pub/WWW/Addressing/Addressing.html.

---

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Use</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>Closes the specified display. <strong>Note:</strong> This keyword is necessary only where strict control over documents is required for security reasons, since by default the server automatically closes open displays when the period specified in the server parameter GW_TIMEOUT has expired. For more information about GW_TIMEOUT see Section 2.5.2, “Creating the Graphics Server cartridge”.</td>
<td>(yes) is the only valid value for the close keyword. For example: sessionid=000001.091032 close=yes</td>
</tr>
</tbody>
</table>

**Note:** Closes the specified display. (yes) is the only valid value for the close keyword. For example: sessionid=000001.091032 close=yes
2

Setting Up the Developer/2000 Server

This section covers these topics:

- Configuring the Forms Server
- Configuring the Reports Server
- Configuring the Graphics Server

2.1 Configuring the Forms Server

When you have installed the software onto your application server, you must configure the Developer/2000 Server, your Web server, and Oracle Web Request Broker\(^1\) in order to deploy Forms applications on the Web.

For instructions, refer to:

- Step 1: Generating and deploying .FMX files
- Step 2: Starting and stopping the Forms Server Listener
- Step 3: Making applications available to end users
- Step 4: Configuring the Forms Client

Note: If, after installation and configuration, you experience problems running Forms applications on the Web, refer to Section 2.2, “Troubleshooting your Forms configuration”.

\(^1\) Required only for cartridge implementation.
2.1.1 Step 1: Generating and deploying .FMX files

Once you have developed the form definition (.FMB file) of a form, and you want to deploy the form on the Web, you must do the following:

1. Generate a Forms Server runtime executable (.FMX file).
   
   You must generate .FMX files on the same platform as the application server on which you will deploy them. For example, if your application server’s operating system is Sun Solaris, you must use the Solaris version of the Forms Generate component to generate .FMX files for deployment on the Web.

2. Deploy the .FMX file on your application server.
   
   You can deploy .FMX files in any directory on your application server, provided you include the appropriate directory path and file name in the cartridge or HTML file that end users access to run the application.
   
   If you provide only the .FMX file name in your application cartridge or HTML file (with no directory path), the Forms Server runtime engine looks in two places for the file:
   
   - **ORACLE_HOME\BIN**
     - The Runtime Engine searches here first.
   
   - **FORMS45_PATH**
     - If the Runtime Engine cannot find the .FMX file in **ORACLE_HOME\BIN**, it then searches the directory tree defined by the **FORMS45_PATH** environment variable (if defined) for the first occurrence of the .FMX file.
     
     You can view and set the **ORACLE_HOME** and **FORMS45_PATH** environment variables.

2.1.2 Step 2: Starting and stopping the Forms Server Listener

Before users can run your Web-enabled Forms applications, you must start a Forms Server Listener process on your application server. The Listener handles Web application requests from end users.

For instructions, refer to:

- Starting a Forms Server Listener process
- Checking the status of the Forms Server Listener
- Stopping a Forms Server Listener process
2.1.2.1 Starting a Forms Server Listener process

On Microsoft Windows NT:
1. On the NT taskbar, choose Start—>Run.
2. Type \<ORACLE_HOME>\bin\f45srv32 port=port_number and press the Enter (or Return) key.
   
   For example: c:\orant\bin\f45srv32 port=5555
   
   A Listener process starts running on the specified port number.

On UNIX:
1. From the UNIX prompt, type cd \$ORACLE_HOME/bin and press the Enter (or Return) key.
2. Type f45srvm port=port_number & and press the Enter (or Return) key.
   
   For example: f45srvm port=4321 &
   
   A Listener process starts running (in the background) on the specified port number.

Notes:
- If you do not specify a port number when you start the Listener process, the process starts on port 9000 by default.
- The port number on which you start the Listener process must match the port number you specify in an application’s static (non-cartridge) HTML file, cartridge HTML file, cartridge parameters, or URL. For more information, refer to Section 2.1.3, “Step 3: Making applications available to end users”.

2.1.2.2 Checking the status of the Forms Server Listener

On Microsoft Windows NT:
1. With your mouse, point to a blank area of the NT taskbar, and display the popup menu.
   
   Note: Most users click the right mouse button to display a popup menu.

2. Choose Task Manager.
3. In the Task Manager, click the Processes tab.

If a Listener process is running, the Task Manager will display a process called F45SRV32.EXE, and multiple occurrences of the F45WEB32.EXE process (one for every active connection).

On UNIX:
- At the UNIX prompt, type `ps -ef | grep f45` and press the Enter (or Return) key.

A list of process IDs will appear on the screen. If the Listener is running, the list will include a process called F45SRVM, and multiple occurrences of the F45RUNW process (one for every active connection, plus one spare connection ready for the next user).

### 2.1.2.3 Stopping a Forms Server Listener process

On Microsoft Windows NT:
1. Check the status of the Forms Server Listener.
2. If the Listener is running, the Task Manager will display a process called F45SRV32.EXE.

On UNIX:
1. Check the status of the Forms Server Listener.
   A list of process IDs will appear on the screen. Note the process ID for the f45srvm process.

2. At the UNIX prompt, type `kill process_ID` and press the Enter (or Return) key.

### 2.1.3 Step 3: Making applications available to end users

Once you have generated and deployed the .FMX files for the Forms applications you want to deploy on the Web, make the applications available to end users by following the instructions in:

- Step 3a: Creating virtual directories on your Web server
- Step 3b: Choosing an HTML file construction method, either dynamic or static.
- Step 3c: Providing URLs of your Web-enabled applications
2.1.3.1 Step 3a: Creating virtual directories on your Web server

- To create pointers to various physical directories on your application server, create virtual directories on your Web server.

For more information about creating virtual directories, refer to your Web Server documentation.

Creating virtual directories offers the following benefits:

- **Simplicity.** When you create an application cartridge or custom HTML file, or use a URL to point to the cartridge or HTML file, you can refer to a virtual directory instead of specifying a lengthy physical directory path.

- **Portability.** If you move files to different directories in the future, you only need to modify the virtual directory settings on your Web server (instead of modifying existing HTML files or cartridges).

Consider creating virtual directories for the following:

- **Applet codebase.** To point to the physical directory where the Forms Client will search for Java class files: ORACLE_HOME/forms45/java (for example, \c:\orant\forms45\java)  
  
  **Note:** Do not set the codebase virtual directory to /ORACLE/.

- **HTML files.** To point to the physical directory where the Web server will search for static (non-cartridge) HTML files, and cartridge HTML files.

- **JAR (Java Archive) files.** To point to the physical directory where the Oracle JAR files (and any custom JAR files) are stored.

Below are examples of virtual directory settings. The directories are provided only as examples, and are not meant as guidelines or requirements, with one exception: the physical directory you define for applet codebase must be ORACLE_HOME/forms45/java. Other than applet codebase, choose physical and virtual directory settings according to your own configuration and needs.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example Physical Directory</th>
<th>Example Virtual Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>applet codebase</td>
<td>c:\orant\forms45\java/</td>
<td>/web_code/</td>
</tr>
<tr>
<td>HTML files</td>
<td>c:\web_forms\html/</td>
<td>/web_html/</td>
</tr>
<tr>
<td>JAR files</td>
<td>c:\orant\forms45\java/</td>
<td>/web_jars/</td>
</tr>
</tbody>
</table>
2.1.3.2 Step 3b: Choosing an HTML file construction method

When an end user first starts a Web-enabled Forms application (by selecting the application’s URL), an HTML file is downloaded from the application server to the user’s Web browser. The HTML file contains all necessary applet tags, parameters, and parameter values required to run the selected application on the Web.

The initial HTML file can be constructed in two ways:

- **Dynamically.** The initial HTML files is dynamically constructed by the Forms Cartridge Handler. The benefit of dynamic HTML file delivery is that you can create a generic cartridge and reuse it (for each of your applications) simply by modifying application URLs.

  To create an application cartridge, you must install the Oracle Web Application Server 3.0 or the Oracle WebServer 2.1 on your application server. Both the Oracle Web Application Server 3.0 and the Oracle WebServer 2.1 include a Web Request Broker component, which provides a framework for cartridges, and manages client connections to them.

  For instructions on implementing dynamically-constructed HTML files, refer to Section 2.1.3.3, “Creating an application cartridge”.

- **Statically.** You simply create a static HTML file, hardcoding all information to the individual application. The benefit of static HTML file delivery is that you do not need to install the Oracle Web Application Server 3.0 or Oracle WebServer 2.1 on your application server; you can use any web server.

  For instructions on implementing static HTML files, refer to Section 2.1.3.5, “Creating a static (non-cartridge) HTML file”.

2.1.3.3 Creating an application cartridge

Create an application cartridge for a Forms application by doing the following:

- Creating and registering your application cartridge

- Creating a base HTML file for the application cartridge

**Note:** The following instructions are specific to Oracle Web Application Server 3.0. If you are using Oracle WebServer 2.1, refer to Section C.1, “Creating an Application Cartridge for Your Forms Server”. If you are using another Web server, refer to your Web server documentation for information about creating application cartridges.
Creating and registering your application cartridge

1. Install Oracle Web Application Server 3.0, if you haven’t already.
   
   Note: When you install the Web Application Server 3.0, accept www as the name of the default Web Listener. You specify the listener name in the Oracle Web Listener Configuration dialog.

2. Start the Web Request Broker listener. For example, type:
   
   owsctl start wrb
   
   For more information about the listener, refer to your Oracle WebServer documentation.

3. Start the Admin listener. For example, type:
   
   owsctl start admin

4. Start the WWW listener. For example, type:
   
   owsctl start www

5. Start your browser.

6. Access the Oracle Web Application Server by navigating to the appropriate URL (http://<name of your machine or IP address>:<port number of Web Server Admin listener>).
   
   For example: http://myserver.com:8888.

   The Oracle Web Application Server appears.

7. Click Web Applications Server Manager to display the Administration page.

8. Click Oracle Web Application Server.

9. Click Cartridge Administration to display the Cartridge Administration page.

10. Click Add New Cartridge.
11. Click **Add New Cartridge with Manual Configuration**, then enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this value...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartridge Name</td>
<td>The cartridge name. For example: <code>web_cart</code></td>
</tr>
<tr>
<td>Object Path</td>
<td>The directory path and file name of the Forms Cartridge Handler.</td>
</tr>
<tr>
<td></td>
<td>- On Microsoft Windows NT: <code>%ORACLE_HOME%\bin\f45webc.dll</code></td>
</tr>
<tr>
<td></td>
<td>- On UNIX: <code>$ORACLE_HOME/lib/f45webc.so</code></td>
</tr>
<tr>
<td>Entry Point</td>
<td><code>form_entry</code></td>
</tr>
<tr>
<td></td>
<td><em>form_entry</em> is the name of the entry point routine in the cartridge shared library.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> <code>form_entry</code> is the required value for Entry Point.</td>
</tr>
<tr>
<td>Minimum # of Instances</td>
<td>The minimum number of users that can simultaneously connect to the cartridge.</td>
</tr>
<tr>
<td>Maximum # of Instances</td>
<td>The maximum number of users that can simultaneously connect to the cartridge.</td>
</tr>
<tr>
<td>Virtual Path (under Virtual Paths)</td>
<td>At runtime, end users pass URLs (HTTP requests) to your Web server. If a request contains the virtual directory associated with your cartridge, the Web server automatically transfers the request to the Web Request Broker. The Web Request Broker in turn starts the cartridge associated with the virtual directory. When defining the Virtual Path, do not include a trailing slash. For example: <code>/web_cart</code></td>
</tr>
<tr>
<td>Physical Path (under Virtual Paths)</td>
<td>The directory path for the Forms Cartridge Handler:</td>
</tr>
<tr>
<td></td>
<td>- On Microsoft Windows NT: <code>%ORACLE_HOME%\bin\</code></td>
</tr>
<tr>
<td></td>
<td>- On UNIX: <code>$ORACLE_HOME/lib/</code></td>
</tr>
</tbody>
</table>

12. Click **Register New Cartridge**.

The Oracle Web Application Server 3.0 registers your cartridge and creates a link for your cartridge in the Web Application Server Home page.

13. Click your cartridge link to display the Cartridge Configuration page.
14. Click **Cartridge specific parameters**, then enter the following parameters and parameter values:

**Note:** Cartridge parameters and parameter values are case-sensitive.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseHTML</td>
<td>Enter the physical directory path and file name for the base cartridge HTML file accessed by the Forms Cartridge at runtime. For example: c:\orant\test. For information on creating a cartridge HTML file, refer to “Creating a base HTML file for the application cartridge”.</td>
</tr>
<tr>
<td>HTMLdelimiter</td>
<td>Enter the one-character delimiter used to denote parameter values in the application’s cartridge HTML file. The delimiter can be any character. The default is % (percent sign).</td>
</tr>
</tbody>
</table>

15. (Optional) Enter the following optional cartridge parameters and parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive</td>
<td>Provide the virtual directory path and file name of any JAR file(s) you want downloaded to end users’ Web browsers at application startup. For more information, refer to Section 2.1.4.4, “Referencing JAR files”.</td>
</tr>
<tr>
<td>Code</td>
<td>Enter the following: oracle.forms.uiClient.v1_4.engine.Main</td>
</tr>
<tr>
<td>Codebase</td>
<td>Enter the name of the virtual directory you defined to point to the physical directory ORACLE_HOME\forms45\java\</td>
</tr>
<tr>
<td>serverApp</td>
<td>Enter the name of the application class you wish to apply to the application. Use application classes for creating application-specific font mapping and icon path settings. For more information, refer to Section B.1, “About Application Classes”.</td>
</tr>
<tr>
<td>serverPort</td>
<td>Enter the number of the port on which the Forms Server Listener process was started. In most cases, the port is 9000 (the default). For more information, refer to Section 2.1.2, “Step 2: Starting and stopping the Forms Server Listener”.</td>
</tr>
<tr>
<td>&lt;forms_param&gt;</td>
<td>Enter valid Forms command-line parameters (such as module) and corresponding values.</td>
</tr>
<tr>
<td>&lt;user_param&gt;</td>
<td>Enter valid user-defined parameters and corresponding values.</td>
</tr>
</tbody>
</table>

16. Click **Modify Cartridge Configuration**.
17. To activate your changes, do the following:
   - Click **Listener**.
   - Click **Stop** to stop the WWW listener.
   - Click **Start** to restart the WWW listener.

### 2.1.3.4 Creating a base HTML file for the application cartridge

To create a base HTML file:

1. Make a copy of **CARTRIDG.HTML** (an Oracle template file located at `ORACLE_HOME/FORMS45/DEMOS/WEBDEMOS`).
2. Rename the copy.
   For example: **cart.html**
3. Open the file in a text editor and modify the text and applet tags as needed.
   For information on the Oracle cartridge HTML file template (including a snapshot, instructions on modifying the template, and examples of customized files), refer to Section A.2, “Working with the Cartridge HTML File Template”.
4. Place the new file in the physical directory that corresponds to the virtual directory you defined for HTML files. **Note:** For information on creating a virtual directory to point to the physical directory where your HTML files are located, refer to Section 2.1.3.1, “Step 3a: Creating virtual directories on your Web server”.

### 2.1.3.5 Creating a static (non-cartridge) HTML file

To deploy non-cartridge Web applications, you must create a static HTML file for each application you deploy on the Web. The easiest way to do this is to modify the static (non-cartridge) HTML file template provided by Oracle.

**To create a static HTML file for an application:**

1. Make a copy of **STATIC.HTML** (an Oracle template file is located at `ORACLE_HOME/FORMS45/DEMOS/WEBDEMOS`).
2. Rename the copy.
   For example, if your application is Order Info: **order.html**
3. Open the file in a text editor and modify the text and applet tags as needed.

   For information on the Oracle static (non-cartridge) HTML file template (including a snapshot, instructions on modifying the template, and examples of customized files), refer to Section A.3, “Working with the Static (Non-Cartridge) HTML File Template”.

4. Place the new file in the physical directory that corresponds to the virtual directory you defined for HTML files. **Note:** For information on creating a virtual directory to point to the physical directory where your HTML files are located, refer to Section 2.1.3.1, “Step 3a: Creating virtual directories on your Web server”.

### 2.1.3.6 Step 3c: Providing URLs of your Web-enabled applications

Once you have created a cartridge or static HTML file for an application, and have deployed the corresponding .FMX files, you must give end users access to the application. To do this, simply provide end users with the URL for the application cartridge or HTML page. End users then contact the URL with their Java-enabled Web browser and run the corresponding application.

**Note:** Another idea is to create an application menu that contains URL links to all of your Web-enabled Forms applications. For more information, see Section 3.1.3, “Creating an HTML application menu”.

The URL you create will differ depending on whether your application is a cartridge or non-cartridge implementation.

**Cartridge URL**

If you created a cartridge for your application, the URL you give to end users must contain the virtual directory associated with the application cartridge, plus parameter values (if required). For example, to announce the availability of its new cartridge-driven Inventory Checking application, General Traders, Inc. gives the following URL to their end users:

```
http://gti.com/web_cart?module=inv&userid=clerk@stock&deptid=90
```

GTI’s URL consists of the following components:

- **Protocol:** http
- **Domain:** gti.com
- **Web server listener port:** 80 (default, implicit)
- Cartridge virtual directory: /web_cart
- Command-line parameters: module (value = inv) and userid (value = clerk@stock)
- User-defined parameter: deptid (value = 90)

Non-cartridge URL

If you created a static HTML page for your application, then the URL you give to end users simply points to the page. For example, to announce the availability of its new Order Tracking application, ABC Corp. might broadcast the following URL:

http://www.abc.com:6666/apps_html/order.html

ABC’s URL consists of the following components:
- Protocol: http
- Domain: www.abc.com
- Web server listener port: 6666 (explicit)
- HTML files virtual directory: /apps_html
- Static HTML file: order.html

2.1.4 Step 4: Configuring the Forms Client

When an end user starts up a Web-enabled Forms application, the Forms Client (and related Java class files) download from the application server to the user’s browser. As the user continues interacting with the application, additional Java class files are downloaded on an as-needed basis.

You can control how class files are downloaded to the user’s browser. There are two methods:

- Incremental. If you select incremental download (the default), only those class files required to render the initial state of the application download at startup. For example, if the initial state of an application includes only a text field and a button, then three Java class files download immediately:
  1. Runform (the Forms Client, downloaded first by default)
  2. CfmTextField (the class file for text fields)
  3. CfmButton (the class file for buttons)
If the end user clicks the button to display an alert, the class file CfmAlert downloads to the client. Note that a network roundtrip is required for each class file downloaded to the user’s machine.

- **Bundled.** With a bundled download, one or more “bundles” of class files are downloaded to the client machine at application startup. The advantage of bundled download is that each bundle downloads in a single network roundtrip. To download class files in bundles, you must reference a JAR file in the application’s cartridge settings or HTML file.

### 2.1.4.1 About JAR files

JAR files are used for aggregating multiple files (Java class files, images, and so on) into one file. Using JAR files increases the speed with which the Forms Client applet and related class files download (and start executing) for two reasons:

- A JAR file downloads in a single HTTP transaction, rather than opening a new connection for each component file.
- JAR file format supports compression, which reduces the size of the downloaded file(s).

At application startup, when Java identifies the JAR in the HTML file, the JAR downloads to the browser and separates into its component files. When the Forms Client requests a new class file during execution, it searches for it in the archives (class files from JAR files) associated with the applet. This search occurs on the client machine. If the applet does not find the class file in the archives on the client machine, it will search for the class file on the application server from which the applet was downloaded.

### 2.1.4.2 About the generic Oracle JAR file

The Oracle Installer automatically installs a generic JAR file (ORACLE_HOME\forms45\java\formsweb.jar), that contains many commonly-used Forms and message class files. The formsweb.jar file includes Runform.class, which is the first class file to be executed at application startup.

### 2.1.4.3 Creating custom JAR files

If the collection of class files in the generic Oracle JAR file does not suit your needs, you can create your own JAR file.
To create a custom JAR file, you must use the JAR utility from Javasoft to create bundles of Forms Java class files. To get the JAR utility and the documentation explaining how to use it, contact Javasoft’s website at:

www.javasoft.com/

If you create your own JAR files, be sure to:

- Locate your JAR file(s) in a physical directory outside of the ORACLE_HOME directory tree.
- Create a virtual directory on your Web server to point to the physical directory where your JAR files are located.
- Include Runform.class inside the JAR file, since it is the first class file executed at application startup. If Java cannot find Runform.class in the class files separated out of your custom JAR file, it will search for Runform.class on the application server. This extra network roundtrip defeats the purpose of JAR files.

2.1.4.4 Referencing JAR files

To reference a JAR file to be downloaded at startup, simply add the ARCHIVES tag to the parameter settings of the application cartridge or HTML file.

Notes:

- If you reference the Oracle JAR file, the virtual directory path of the JAR file must be the virtual directory defined for the applet codebase.
- If you reference custom JAR files, the virtual directory path of the JAR files must be the virtual directory defined for custom JAR files.
- You can reference multiple JAR files in a single ARCHIVES tag. Simple separate each JAR file reference with a “+” (addition sign).
- The order in which multiple JAR files are downloaded is determined by the order (left to right) in which they are referenced after the ARCHIVES tag.
2.1.4.5 Example of adding the ARCHIVES tag to an application’s static HTML file:

```html
<applet codebase="/web_code/"
   code="oracle.forms.uiClient.v1_4.engine.Runform">
   <param name="archives" value="/..//web_code/formsweb.jar + 
    /..//acme_jars/basic.jar + 
    /..//acme_jars/advanced.jar">
   <param name="serverPort" value="9000">
   <param name="serverArgs" value="module=orders">
</applet>
```

2.1.5 Setting up load balancing

The load balancing feature allows you to dynamically balance the load of multiple Forms Servers across systems. Load balancing directs a Forms Client to connect to a Forms Server running on the least-loaded system that is available. Which system is least loaded is determined by a count of the total number of processes running on that system.
Following is an example of a possible load balancing configuration:

*The Forms Cartridge asks the Metrics Server for the name of the least-loaded system. Then the Forms Cartridge creates an HTML file, including the host name, and sends it back to an end user’s Web browser.*

*Figure 2–1 Load balancing configuration*

The Metrics Server and Metrics Clients are services on Windows NT and they are daemons on UNIX. The software for these services is included when you install.
Form Builder. The Metrics Server and Metrics Clients do not become active until you complete the steps listed in Table 2–1, “Load balancing roadmap”.

When you use load balancing, the following events occur:

1. Metrics Clients periodically send load information to the Metrics Server. This load information is based on the total number of processes running on each Metrics Client.

2. End users access a URL pointing to a Forms Cartridge.

3. The Forms cartridge asks the Metrics Server for the name of the least-loaded system that is available.

4. The Forms cartridge dynamically creates an HTML file with the name of the least-loaded system specified as the system on which to run the Forms Server and returns that HTML file to an end user’s Web browser.

5. The user’s Web browser then requests the Java applet to be downloaded from the host specified in the HTML file.

6. The Java applet sends a request to the Forms Server Listener asking for a particular Form Builder application (that is, an FMX).

7. The listener contacts a Forms Server Runtime Engine. (The listener maintains a pool of available Runtime Engines to minimize application startup delays.) Each active user receives a dedicated Runtime Engine.

8. The listener establishes a direct socket connection with the Runtime Engine, and sends the socket information to the Java applet. The Java applet then establishes a direct socket connection with the Runtime Engine. The Java applet and the Runtime Engine now communicate directly, freeing the listener to accept startup requests from other end users. (At this point, neither the Web server nor the Forms listener is involved in the communication between the applet and the Runtime Engine.) The Java applet displays the application’s user interface in an applet window outside the main window of the end user’s Web browser.

9. As in a client-server implementation, the Runtime Engine communicates directly with the database through SQL*Net or ODBC, depending on the data source.

Metrics Clients continue to send load information to the Metrics Server and all new service requests are routed based on that information.

Note: If the Metrics Server is unavailable, at Step 3 the Forms cartridge will not get any information back about which is the least-loaded system. The Forms cartridge
will then, instead, specify the value of the parameter MetricsServerErrorURL for the end user’s Web browser to request. The user does not necessarily know this is happening as the redirect is behind the scenes from the user’s viewpoint. Refer to Section 2.1.5.1, “Configuring your Web cartridge for load balancing”, for information about the MetricsServerErrorURL parameter.

Requirements
Following are the requirements for using the Forms Server load balancing feature:

- Metrics Server on one system
  - Forms Server (includes the Metrics Server and the Forms cartridge)
    - **Note:** All Forms Servers must listen on the same port number on each of the systems.
  - Oracle WebServer (release 2.1) or Web Application Server (release 3.0)
- Metrics Clients on all systems running Forms Server
  - Forms Server (includes the Metrics Client)
    - **Note:** All Forms Servers must listen on the same port number on each of the systems.
  - Any Web server
Roadmap for using your load balancing environment

Following is the sequence of steps to take to set up and use your load balancing environment:

Table 2–1 Load balancing roadmap

<table>
<thead>
<tr>
<th>Step</th>
<th>Refer to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Configure your load balancing Web cartridge. Section 2.1.5.1, “Configuring your Web cartridge for load balancing”</td>
</tr>
<tr>
<td>2.</td>
<td>Install your Metrics Server and Metrics Client. Section 2.1.5.2, “Installing the Metrics Server and Clients”</td>
</tr>
<tr>
<td>3.</td>
<td>Modify your cartridge base HTML file. Section 2.1.5.3, “Modifying your cartridge base HTML file”</td>
</tr>
<tr>
<td>4.</td>
<td>Start your Metrics Server. Section 2.1.5.4, “Starting the Metrics Server”</td>
</tr>
<tr>
<td>5.</td>
<td>Start your Oracle Web Server (on the same system as your Metrics Server). Your Oracle WebServer or Web Application Server documentation.</td>
</tr>
<tr>
<td>6.</td>
<td>Start your Metrics Clients. Section 2.1.5.5, “Starting the Metrics Clients”</td>
</tr>
<tr>
<td>7.</td>
<td>Start the httpd/Web listener on each Metrics Client system. Your Oracle WebServer or Web Application Server documentation or the vendor’s documentation for third-party Web listeners.</td>
</tr>
</tbody>
</table>

The following sections describe the steps for installing, configuring, and setting up load balancing for your Forms Server.

2.1.5.1 Configuring your Web cartridge for load balancing

First do the steps in Section 2.1.3.3, “Creating an application cartridge”.

Then:

1. Access the Oracle Web Application Server by navigating to the appropriate URL (http://<name of your machine or IP address>:<port number of Web Server Admin listener>).

   For example: http://myserver.com:8888.

   The Oracle Web Application Server appears.

2. Click Web Applications Server Manager to display the Administration page.

3. Click Oracle Web Application Server.
4. Click **Cartridge Administration** to display the Cartridge Administration page.

5. Click your Forms Web cartridge.

6. Click **Cartridge specific parameters**, and then enter the following parameters and parameter values which are used by the Forms Server for load balancing:

   **Note:** Cartridge parameters and parameter values are case sensitive.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetricsServerHost</td>
<td>Enter the name of the host (system) where the Metrics Server is running. This parameter is required if load balancing is used. For example:</td>
</tr>
<tr>
<td></td>
<td>system1.company.com</td>
</tr>
<tr>
<td>MetricsServerPort</td>
<td>Enter the port number on which the Metrics Server will be listening for requests from the cartridge for the least-loaded system. This parameter is required if load balancing is used and its value must be different from the Forms Listener value. For example:</td>
</tr>
<tr>
<td></td>
<td>70000</td>
</tr>
<tr>
<td>MetricsServerErrorURL</td>
<td>Enter the URL to which the Web Client will be directed if the Metrics Server does not respond with the name of the least-loaded system within the MetricsTimeout period. This parameter is required if load balancing is used. The URL can:</td>
</tr>
<tr>
<td></td>
<td>■ Point to a Forms Web cartridge on another host, which will then automatically service the request. For example:</td>
</tr>
<tr>
<td></td>
<td><a href="http://altsys.company.com/web_cart?module=inv&amp;userid=clerk@stock&amp;deptid=90">http://altsys.company.com/web_cart?module=inv&amp;userid=clerk@stock&amp;deptid=90</a></td>
</tr>
<tr>
<td></td>
<td>■ Point to a static or dynamic HTML page on any Web server. This page can be used for giving help and instructions to users when the Metrics Server goes down and load balancing is disabled. For example:</td>
</tr>
</tbody>
</table>
7. Click Modify Cartridge Configuration.
8. To activate your changes, do the following:
   1. Click Listener.
   2. Click Stop to stop the WWW listener.
   3. Click Start to restart the WWW listener.

   **Note:** If you plan to use any of your Metrics Clients as backup Metrics Servers, then you need to do these steps for those Metrics Clients as well. Otherwise, you only need to do these configuration steps for the Metrics Server.

### 2.1.5.2 Installing the Metrics Server and Clients
Following are the steps to install the Metrics Server and Metrics Clients.

**Installing the Metrics Server**
On Windows NT, run the following command to install the Metrics Server as an NT Service:

```
d2ls1032 -install
```

To deinstall the Metrics Server service:

```
d2ls1032 -remove
```

On UNIX, the Metrics Server runs as a daemon so there is no installation required. The name of the Metrics Server executable on UNIX is `d2ls10`. 

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetricsTimeout</td>
<td>(Optional) Enter the timeout value, in seconds, for the Forms cartridge to use while making requests to the Metrics Server for the name of the least-loaded system. If the cartridge does not receive a response in that time interval, it replaces the %LEASTLOADEDHOST% entry in the HTML file with the MetricsServerErrorURL value. The MetricsTimeout parameter is optional. If not specified, the default value is 30 seconds. For example:</td>
</tr>
<tr>
<td></td>
<td>35</td>
</tr>
</tbody>
</table>
Installing Metrics Clients
On Windows NT, run the following command on each system to install each Metrics Client as an NT Service:

d2lc1032 -install

To deinstall the Metrics Client service:

d2lc1032 -remove

On UNIX, the Metrics Client runs as a daemon so there is no installation required. The name of the Metrics Client executable on UNIX is d2lc10.

2.1.5.3 Modifying your cartridge base HTML file
%LEASTLOADEDHOST% is a fixed-name place holder in the base HTML file. This place holder is replaced dynamically by the cartridge with the name of the least-loaded system. The cartridge gets this information from the Metrics Server at runtime.

Refer to Section 2.1.3.4, “Creating a base HTML file for the application cartridge”, for information about creating your base HTML file.

Refer to Section A.2.2, “Modifying the cartridge HTML file template”, for information about using the %LEASTLOADEDHOST% parameter in your base HTML file.

2.1.5.4 Starting the Metrics Server

Starting the Metrics Server on Windows NT
1. Click Start—>Settings—>Control Panel.
2. Double-click Services.
4. Enter values for the following startup parameters:

   MetricsClientToServerPort# MetricsServerToCartridgePort#
   [max_no_MetricsClients]

where:

   ■ MetricsClientToServerPort# is the port on the Metrics Server where the Metrics Clients will connect and will send their load information.
- \textit{MetricsServerToCartridgePort}# is the port on the Metrics Server which the cartridges will use to query the Metrics Server for information. This value corresponds to the \textit{MetricsServerPort} property, as described in “Configuring your Web cartridge for load balancing”.

- \textit{max\_no\_MetricsClients} is the maximum number of Metrics Clients that will be running and sending load information to the Metrics Server. This parameter is optional. The default value is 1000.

5. Click \textbf{Start} to start the service.

**Starting the Metrics Server on UNIX**

Enter the following command to start your Metrics Server on UNIX:

\begin{verbatim}
d2ls10 MetricsClientToServerPort# MetricsServerToCartridgePort#
[max_no_MetricsClients]
\end{verbatim}

where:

- \textit{MetricsClientToServerPort}# is the port on the Metrics Server where the Metrics Clients will connect and will send their load information.

- \textit{MetricsServerToCartridgePort}# is the port on the Metrics Server which the cartridges will use to query the Metrics Server for information.

- \textit{max\_no\_MetricsClients} is the maximum number of Metrics Clients that will be running and sending load information to the Metrics Server. This parameter is optional. The default value is 1000.

2.1.5.5 **Starting the Metrics Clients**

Do these steps for each system where a Metrics Client will run.

**Starting the Metrics Clients on Windows NT**

1. Click \textbf{Start}—>Settings—>Control Panel.
2. Double-click \textbf{Services}.
3. Choose \textbf{D2LC Client Service}.
4. Enter values for the following startup parameters:

\begin{verbatim}
MetricsServerHostName MetricsClientToServerPort# MetricsClientLocalPort#
[ScaleFactor]
\end{verbatim}
where:

- `MetricsServerHostName` is the name of the system where the Metrics Server is running.
- `MetricsClientToServerPort#` is the port on the Metrics Server where the Metrics Clients will connect.
- `MetricsClientLocalPort#` is the port on the Metrics Client which the Client will use to query the Metrics Server for information.
- `ScaleFactor` is an optional parameter that allows you to reduce the imbalances resulting from varying capacities of Metrics Clients. Since Forms Server load balancing is based only on total number of processes running on each Metrics Client, a system that appears to be the least-loaded system may not necessarily be the best place to run a new process. You should assign a higher value for `ScaleFactor` for your lower capacity systems. The default value for `ScaleFactor` is 1 for UNIX and 4 for Windows NT.

5. Click **Start** to start the service.

**Starting the Metrics Clients on UNIX**

Enter the following command to start each Metrics Client that will run on UNIX:

```
d2lc10 MetricsServerHostName MetricsClientToServerPort# MetricsClientLocalPort# [ScaleFactor]
```

where:

- `MetricsServerHostName` is the name of the system where the Metrics Server is running.
- `MetricsClientToServerPort#` is the port on the Metrics Server where the Metrics Clients will connect.
- `MetricsClientLocalPort#` is the port on the Metrics Client which the Client will use to query the Metrics Server for information.
- `ScaleFactor` is an optional parameter that allows you to reduce the imbalances resulting from varying capacities of Metrics Clients. Since Forms Server load balancing is based only on total number of processes running on each Metrics Client, a system that appears to be the least-loaded system may not necessarily be the best place to run a new process. You should assign a higher value for `ScaleFactor` for your lower capacity systems. The default value for `ScaleFactor` is 1 for UNIX and 4 for Windows NT.
Refer to Section 2.1.3.6, “Step 3c: Providing URLs of your Web-enabled applications”, for information about giving end users access to the application.

### 2.2 Troubleshooting your Forms configuration

If you experience trouble running Forms on the Web, check the table below for common symptoms and remedies:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>You cannot run Web-enabled Forms applications with a non-Java-enabled Web browser.</td>
<td>If you are not sure your Web browser is Java-enabled, check the network preferences of your Web browser. The Enable Java and Enable JavaScripts check boxes must be set to checked.</td>
</tr>
<tr>
<td>You see the error message Cannot bind to port 9000 when you try to start the Forms Server.</td>
<td>It is likely that another process is using the port. It could be another occurrence of the Forms Server, so check that it is not already running (see Section 2.1.2, “Step 2: Starting and stopping the Forms Server Listener”). If you just stopped the Forms Server, it may take a minute or two for existing connections to port 9000 to reopen.</td>
</tr>
<tr>
<td>The Forms Client does not download to your Web browser.</td>
<td>Check that you have not defined a virtual directory called /ORACLE/ to point to the Oracle Java class files (codebase).</td>
</tr>
</tbody>
</table>
## Troubleshooting your Forms configuration

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
</table>
| The HTML page and applet download at startup, and the applet starts running but nothing else seems to happen. | Check the following:  
- First, check that the Forms Client indeed is running; if it is, you should see a message in the status bar of your Web browser: `applet oracle.forms.uiClient_v1_4.engine.Runform running`  
If you see this message, but your application still does not appear, check the following:  
1. Make sure the Forms Server and your Web server are installed on the same application server. Due to a current Java restriction, they must be installed on the same server.  
2. Check your application cartridge or HTML file to make sure you specified a valid directory path and file name for the `.FMX` file. You must use a physical directory path, not a virtual directory path.  
3. Try setting a preference in your Web browser to display the Java console. This allows you to view runtime Java error messages. |
| The Forms Server seems to ignore the user ID, password, and database SID parameter values you pass in your application cartridge or HTML file. | Make sure you preface the values with the parameter `userid=`. For example:  
`userid=scott/tiger@inventory`. |
| The Forms Server seems to not pick up your variable changes. | Stop and restart the Forms Server Listener. For more information, see Section 2.1.2, “Step 2: Starting and stopping the Forms Server Listener”. |
| You experience problems when using a security firewall, and you are using a proxy server to access anything outside the firewall. | Make sure your proxies are set to manual configuration. |
Troubleshooting your Forms configuration

You experience trouble connecting to a local database.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>It could be a result of the following:</td>
<td></td>
</tr>
<tr>
<td>■ If you do not specify a SQL*Net v2 connect string, you will receive errors. The Forms Server runtime engine will not accept connect strings of type LOCAL, TWO_TASK, and so on.</td>
<td></td>
</tr>
<tr>
<td>■ If you are using a SQL*Net v2 connect string and you still cannot connect to the database, make sure the Forms Server Listener is running; on most installations, the Listener is not automatically restarted after a reboot. For information on starting the Listener, see Section 2.1.2.1, “Starting a Forms Server Listener process”.</td>
<td></td>
</tr>
<tr>
<td>■ You must have the valid connect string in the TNSNAMES.ORA file on your application server, not on your client machine. The application logic is running on an application server, not on end users client machines.</td>
<td></td>
</tr>
</tbody>
</table>

You experience unpredictable behavior after modifying the CLASSPATH environment variable

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing the setting of the CLASSPATH environment variable—on your application server or on an end user’s machine—can produce unpredictable results. Setting the variable to a directory that overlaps with the directory tree where Forms Java class files are located can cause file name overlap.</td>
<td></td>
</tr>
</tbody>
</table>

There appears to be several unused processes running on the server.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall that for each user running a Web-enabled Forms application, a Forms Server runtime process (f45run32.exe or f45runw) starts on your application server. Each runtime process should end when the user exits the application, but will remain on the server if a user exits the browser without cleanly exiting the application. To cleanly exit the application, use the menu or the [Exit/Cancel] key function, then exit the browser.</td>
<td></td>
</tr>
</tbody>
</table>

You see the error message Initializing Load Balancing component [n].

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is an internal error generated by the load balancing cartridge. When you get this error, make note of the number in square brackets and contact Oracle Technical Support.</td>
<td></td>
</tr>
</tbody>
</table>
2.3 Configuring the Reports Server

When you install Developer/2000 Reports on your application server, the Reports Server, Web Cartridge, and Web CGI, are copied along with other required files. For the Web, you must configure the Reports Server and either the Reports Web Cartridge or Web CGI.

Note: For non-Web environments, you can simply configure the Reports Server by itself, without the Reports Web Cartridge or Web CGI.

This section covers these topics:

- Section 2.3.1, “Step 1: Setting up the Reports Server”
- Section 2.3.2, “Step 2: Setting up the Reports Web Cartridge or Web CGI”
- Section 2.3.3, “Step 3: Specifying URL run requests”
- Section 2.3.4, “Step 4 (Optional): Using the URL mapping feature”
- Section 2.3.5, “Step 5: Deploying reports on the Web”

2.3.1 Step 1: Setting up the Reports Server

The setup of the Reports Server varies between platforms. Refer to R25MTSUS.HTM for more detailed information on the Reports Server.

2.3.1.1 Setting up for Windows 95

1. Install Reports 3.0 in its entirety or just the Reports Server, Web Cartridge, and Web CGI.

2. Add a line of the following form to $ORACLE_HOME\network\admin\tnsnames.ora:

   `<tnsname> = (ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>)(PORT=<portnum>))`

   where `<tnsname>` is the name of the Reports Server instance. If you are using a sqlnet.ora file and default_domain is specified, remember to fully qualify the Reports Server TNS name with the default domain. `<hostname>` is the IP address of the machine. `<portnum>` is the port number to which the Reports Server is listening. Below is an example entry in tnsnames.ora:

   `repserver.world=(ADDRESS=(PROTOCOL=tcp) (HOST=12.34.56.78) (PORT=1949))`

3. Run the following command line to start the Reports Server:

   `r25mts32 name=<tnsname> [minengine=<number> maxengine=<number> initengine=<number> maxidle=<number>]`
Usage Notes

- To run the server in a Web environment, you need to set up the Reports Web CGI or cartridge.
- To enable reports to be run from Oracle InterOffice, ensure that the RWICLI environment variable points to the thin client executable for your platform, R25CL132.EXE on Windows 95/NT and r25clim on UNIX platforms.
- When the Reports Server is running, an icon will appear in the notification area in the task bar. Double-clicking the icon displays a window showing the basic server status.
- You should not log off of Windows 95 when the Reports Server is running.

2.3.1.2 Setting up for Windows NT as a service

On Windows NT, the Reports Server is best run as an NT service. As with any other service, you can configure the Reports Server service to be started manually or automatically when the system is brought up.

1. Install Reports 3.0 in its entirety or just the Reports Server, Web Cartridge, and Web CGI.

2. Add a line of the following form to $ORACLE_HOME/network/admin/tnsnames.ora:

   `<tnsname> = (ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>) (PORT=<portnum>))`

   where `<tnsname>` is the name of the Reports Server instance. If you are using a sqlnet.ora file and default_domain is specified, remember to fully qualify the Reports Server TNS name with the default domain. `<hostname>` is the IP address of the machine. `<portnum>` is the port number to which the Reports Server is listening. Below is an example entry in tnsnames.ora:

   `repserver.world=(ADDRESS=(PROTOCOL=tcp)(HOST=12.34.56.78)(PORT=1949))`

3. Install the Reports Server as an NT service by running the following command line:

   `r25mts32 -install <tnsname> tcpip`

   Following `<tnsname>` are the names of NT services upon which the Report Server depends (for example, tcpip).
4. Go to Control Panel, Services. Look for the entry “Developer/2000 Reports Multi-tier Server for <tnsname>.” Click on Startup. From this startup dialog, select “This Account” on the Log On As section and type in your operating system username and password. This specifies that the Reports Server is run as you. In addition, you can also set the service to be started automatically (when the system is booted up) or manually.

5. To start the service, go to Control Panel, Services, select the service, optionally type command line arguments in Startup Parameters, and click on the Start button.

6. A configuration file is created when the Reports Server is first started up. The file is $ORACLE_HOME\report25\server\<tnsname>.ora. Note that if you manually edit the file, the changes are not picked up by the Reports Server until you shut it down and restart it.

Usage Notes

■ To enable reports to be run from Oracle InterOffice, ensure that the RWICLI environment variable points to the thin client executable for your platform, R25CLI32.EXE on Windows 95/NT and r25clim on UNIX platforms.

■ You can run multiple Reports Servers on one machine. Just make sure that you specify different port numbers.

■ When the Server is started up, the configuration file will be read, if it exists, or created, if it does not exist. The four optional server command line arguments (minengine, maxengine, initengine, and maxidle) can be specified in the Startup Parameters in the Control Panel Services window to override the settings in the configuration file.

■ Make sure printers are accessible by the service.

■ When running the Reports Server as a service, there is a known problem with Windows NT not sending report output to mail (DESTYPE=MAIL). Microsoft plans to fix this in Microsoft Exchange Server 5.0. In the meantime, you can solve this problem by running the Reports Server in the foreground by running the following command: R25MTS32.EXE -listen <tnsname>.

■ If the service is installed to be run under a user other than SYSTEM, make sure that the user account has the Password Never Expires option selected in the User Manager and has membership in the appropriate groups to run the Reports Server and access the report files. Also make sure that the user can log on to a service. Go to Administration Tools User Manager, Policies, User Rights. Check Show Advanced User Rights, choose Log on a Service, and click the Add...
button. Also make sure the default printer is set and the user has at least print permission on the printer.

- Note that by default, the SYSTEM user does not have access to printers. In order for the Reports Server to run properly, it must have access to printers. Given this, you must either set up a separate user to run the Reports Server or give the SYSTEM user access to printers.

- If the service is installed to be run under the SYSTEM user, an icon will appear on the console desktop.

- Since network drives are mapped to a drive letter on a per-user basis, these mappings are no longer in effect when the NT user logs off. The Reports Server must not refer to these drives through their drive letters. Instead you should use UNC pathnames (for example, \sales\documents\reports). This applies to Reports Server parameters, CGI command mappings, and each hard-coded path name in each report being run.

- To uninstall the Reports Server service, go to Control Panel, Services and stop the service. Then run the following command line:

  r25mts32 -uninstall <tnsname>

### 2.3.1.3 Setting up for Windows NT as a non-service

In general, it is best to install the Reports Server as an NT service, but it can also be installed as a non-service.

1. Install Reports 3.0 in its entirety or just the Reports Server, Web Cartridge, and Web CGI.

2. Add a line of the following form to

$ORACLE_HOME\network\admin\tnsnames.ora:

  <tnsname> = (ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>)(PORT=<portnum>))

where <tnsname> is the name of the Reports Server instance. If you are using a sqlnet.ora file and default_domain is specified, remember to fully qualify the Reports Server TNS name with the default domain. <hostname> is the IP address of the machine. <portnum> is the port number to which the Reports Server is listening. Below is an example entry in tnsnames.ora:

  repserver.world= (ADDRESS=(PROTOCOL=tcp)(HOST=12.34.56.78)(PORT=1949))
3. Run the following command line to start the Reports Server:

```bash
r25mts32 -listen name=<tnsname> [minengine=<number> maxengine=<number>
initengine=<number> maxidle=<number>]
```

For more detailed information about the Reports Server command line, refer to R25MTSUS.HTM.

**Usage Notes**

- To enable reports to be run from Oracle InterOffice, ensure that the RWICLI environment variable points to the thin client executable for your platform, R25CLI32.EXE on Windows 95/NT and r25clim on UNIX platforms.

### 2.3.1.4 Setting up for UNIX

1. Install Reports 3.0 in its entirety or just the Reports Server, Web Cartridge, and Web CGI.
2. Add `$ORACLE_HOME/bin` to the `PATH` environment variable.
3. Set TNS_ADMIN to the location of `tnsnames.ora` (for example, `$ORACLE_HOME/network/admin`).
4. Add a line of the following form to `$ORACLE_HOME/network/admin/tnsnames.ora`:

   ```
   <tnsname> = (ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>)(PORT=<portnum>))
   ```

   where `<tnsname>` is the name of the Reports Server instance. If you are using a `sqlnet.ora` file and `default_domain` is specified, remember to fully qualify the Reports Server TNS name with the default domain. `<hostname>` is the IP address of the machine. `<portnum>` is the port number to which the Reports Server is listening. Below is an example entry in `tnsnames.ora`:

   ```
   repserver.world=(ADDRESS=(PROTOCOL=tcp) (HOST=12.34.56.78) (PORT=1949))
   ```

5. Run the following command line to the background to start the Reports Server:

```bash
r25mtsm name=<tnsname> [minengine=<number> maxengine=<number>
initengine=<number> maxidle=<number>]
```
Usage Notes

- To run the server in a Web environment, you need to set up the Reports Web Cartridge or Web CGI.
- To enable reports to be run from Oracle InterOffice, ensure that the **RWICLI** environment variable points to the thin client executable for your platform, **R25CLI32.EXE** on Windows 95/NT and **r25clim** on UNIX platforms.

### 2.3.2 Step 2: Setting up the Reports Web Cartridge or Web CGI

If you are using the Oracle Web Application Server 3.0 or Oracle WebServer 2.1, you can use the Reports Web Cartridge to handle job requests for the Reports Server. Otherwise, you can use the Reports Web CGI to handle job requests for the Reports Server. Refer to **R25OWS.HTM** for more detailed information about the Reports Web Cartridge. Refer to **R25CGI.HTM** for more detailed information about the Reports Web CGI.

#### 2.3.2.1 Setting up the Reports Web Cartridge

**Note:** The following instructions are specific to Oracle Web Application Server 3.0. If you are using Oracle WebServer 2.1, refer to Section C.2, “Setting Up the Reports Web Cartridge”. If you are using another Web server, refer to your Web server documentation for information about creating application cartridges.

1. Install Reports 3.0 in its entirety or just the Reports Server, Web Cartridge, and Web CGI.
2. Install Oracle Web Application Server 3.0.
   
   **Note:** When you install the Web Application Server, accept **WWW** as the name of the default Web Listener. You specify the listener name in the Oracle Web Listener Configuration dialog.
3. Start the Web Request Broker listener. For example, type:
   ```
   owsc tl start wrb
   ```
   For more information about the listener, refer to your Oracle WebServer documentation.
4. Start the Admin listener. For example, type:
   ```
   owsc tl start admin
   ```
5. Start the WWW listener. For example, type:
   ```
   owsc tl start www
   ```
6. Start your browser.
7. Access the Oracle Web Application Server by navigating to the appropriate URL (http://<name of your machine or IP address>:<port number of Web Server Admin listener>).

For example: http://myserver.com:8888.

The Oracle Web Application Server appears.

8. Click Web Applications Server Manager to display the Administration page.

9. Click Oracle Web Application Server.

10. Click Cartridge Administration to display the Cartridge Administration page.

11. Click Add New Cartridge.

12. Click Add New Cartridge with Manual Configuration, then enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this value...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartridge Name</td>
<td>The cartridge name. For example: r25ows</td>
</tr>
<tr>
<td>Object Path</td>
<td>The directory path and file name of the Reports Web Cartridge.</td>
</tr>
<tr>
<td></td>
<td>■ On Microsoft Windows NT: %ORACLE_HOME%\bin\r25ows32.dll</td>
</tr>
<tr>
<td></td>
<td>■ On UNIX: $ORACLE_HOME/r25owsm.so</td>
</tr>
<tr>
<td>Entry Point</td>
<td>r25ows_start</td>
</tr>
<tr>
<td></td>
<td>r25ows_start is the name of the entry point routine in the cartridge shared library.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> r25ows_start is the required value for Entry Point.</td>
</tr>
<tr>
<td>Minimum # of Instances</td>
<td>The minimum number of users that can simultaneously connect to the cartridge.</td>
</tr>
<tr>
<td>Maximum # of Instances</td>
<td>The maximum number of users that can simultaneously connect to the cartridge.</td>
</tr>
</tbody>
</table>
13. Click Register New Cartridge.

The Web Application Server registers your cartridge and creates a link for your cartridge in the Web Application Server Home page.

14. Click your cartridge link to display the Cartridge Configuration page.

15. Click Cartridge specific parameters, then enter the following parameters and parameter values:

   **Note:** Cartridge parameters and parameter values are case-sensitive.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTS25_WEBLOC</td>
<td>For example, on Windows NT:</td>
</tr>
<tr>
<td></td>
<td>/cache/</td>
</tr>
<tr>
<td>REPORTS25_WEBLOC_TRANSLATED</td>
<td>For example, on Windows NT:</td>
</tr>
<tr>
<td></td>
<td>c:\orant\mycache</td>
</tr>
</tbody>
</table>

16. (Optional) Set up optional cartridge parameters and environment variables to obtain further functionality from R25OWS.

17. Click Modify Cartridge Configuration.
18. To activate your changes, do the following:
   ■ Click **Listener**.
   ■ Click **Stop** to stop the WWW listener.
   ■ Click **Start** to restart the WWW listener.

2.3.2.2 Setting Up the Reports Web CGI for Windows 95/NT

1. Install Reports 3.0 in its entirety or just the Reports Server, Web Cartridge, and Web CGI.
2. Copy the file **R25CGI32.EXE** (located in `$ORACLE_HOME\BIN`) into your Web server script directory.
3. Define the environment variables `REPORTS25_WEBLOC` and `REPORTS25_WEBLOC_TRANSLATED`.

2.3.2.3 Setting Up the Reports Web CGI for UNIX

1. Install Reports 3.0 in its entirety or just the Reports Server, Web Cartridge, and Web CGI.
2. Copy the file **r25cgim** (located in `$ORACLE_HOME/bin`) into your Web server script directory.
3. Set up a shell script in the Web Server script directory to define the environment variables listed below and invoke **r25cgim**.

**R25CGI Environment Variables:** `REPORTS25_WEBLOC`, `REPORTS25_WEBLOC_TRANSLATED`

**Other Required Environment Variables:** `DISPLAY`, `ORACLE_HOME`, `PATH`, `ORACLE_SID`, `TNS_ADMIN`, `TWO_TASK`, `REPORTS25_PATH`, `LD_LIBRARY_PATH`
Example script:

```bash
#!/bin/sh

# cgi-specific variables
REPORTS25_WEBLOC=myvirtualdir
REPORTS25_WEBLOC_TRANSLATED=/blah/wwwroot/myactualdir

# Other required variables
DISPLAY=mymachine:0.0; export DISPLAY
ORACLE_HOME=/blah/myoracledir; export ORACLE_HOME
PATH=.:${ORACLE_HOME}/bin; export PATH
TNS_ADMIN=${ORACLE_HOME}/network/admin; export TNS_ADMIN
TWO_TASK=mytwotask; export TWO_TASK
LD_LIBRARY_PATH=${ORACLE_HOME}/lib; export LD_LIBRARY_PATH

# Here’s the call to the cgi executable
./r25cgim
```

### 2.3.2.4 Cartridge Parameters and Environment Variables

Parameters and environment variables are used to further configure the Reports Web Cartridge and Web CGI.

- Section 2.3.2.5, “ORACLE_HOME”
- Section 2.3.2.6, “LD_LIBRARY_PATH”
- Section 2.3.2.7, “REPORTS25_WEBLOC”
- Section 2.3.2.8, “REPORTS25_WEBLOC_TRANSLATED”
- Section 2.3.2.9, “REPORTS25_OWSHELP”
- Section 2.3.2.10, “REPORTS25_OWSMAP”
- Section 2.3.2.11, “REPORTS25_OWSDIAGBODYTAGS”
- Section 2.3.2.12, “REPORTS25_OWSDIAGHEADTAGS”
- Section 2.3.2.13, “REPORTS25_OWSPATHONLYURL”
- Section 2.3.2.14, “REPORTS25_OWSNODIAG”
- Section 2.3.2.15, “REPORTS25_CGIHLP”
- Section 2.3.2.16, “REPORTS25_CGIMAP”
- Section 2.3.2.17, “REPORTS25_CGIDIAGBODYTAGS”
Section 2.3.2.18, “REPORTS25_CGIDIAGHEADTAGS”
Section 2.3.2.19, “REPORTS25_CGINODIAG”
Section 2.3.2.20, “REPORTS25_CGIPATHONLYURL”
Section 2.3.2.21, “USER_NLS_LANG or NLS_LANG”

Note: The way these parameters/variables are set varies. On Windows 95/NT, they are set in the System Registry. On UNIX, they are set with the `setenv` command. For the Reports Web Cartridge, they are set as parameters (that is, in the Oracle Web Application Server 3.0).

2.3.2.5 ORACLE_HOME
Specifies the file directory where all Oracle software is installed (for example, `c:\orant\` on Windows NT). The location of Oracle shared libraries, NLS resource and message files is derived from this variable. On Windows, it is usually set up during the installation process.

2.3.2.6 LD_LIBRARY_PATH
On UNIX, specifies directory path to search for shared libraries.

2.3.2.7 REPORTS25_WEBLOC
Specifies where the Web server looks for the Reports Server output (file cache). It is a virtual directory, as defined in the Web server configuration file. It may be mapped directly to the Reports Server file cache directory (defined in the `.ora` file for the Reports Server being used) only if that directory is mapped from the Web server machine. `REPORTS25_WEBLOC` is usually mapped to the physical directory defined by the `REPORTS25_WEBLOC_TRANSLATED` parameter, which is valid regardless of whether the Reports Server and the Web server share the same file system.

If this parameter is not set, R25OWS or R25CGI will not display HTML/PDF output in the client browser (“Cannot access report output” message will result), although the report will run and output will be produced.

Examples:

Note: cachedir is set in the Reports Server configuration file. `REPORTS25_WEBLOC` and `REPORTS25_WEBLOC_TRANSLATED` are set as parameters (that is, in the Oracle Web Application Server 3.0) for the Reports Web Cartridge and as environment variables for the Reports Web CGI.
cachedir="c:\orant\report25\server\cache"
reports25_webloc="/"
reports25_webloc_translated="C:\orant\ows21\new"

The cache directory of the Reports Server is set to the default value. The REPORTS25_WEBLOC variable means that a Web listener is using "/" as a virtual directory path. The REPORTS25_WEBLOC_TRANSLATED variable sets the physical directory name of the web listener's default virtual root directory.

cachedir="C:\servercache"
reports25_webloc="/repout"
reports25_webloc_translated="C:\repdir"

This case is similar to the first example, except that the settings are not the default directories.

cachedir="C:\orant\ows21\new"
reports25_webloc="/"

In this case, the Web listener and the Reports Server cache share the same directory. You should not set REPORTS25_WEBLOC_TRANSLATED in this situation because it will cause problems in retrieving HTML output.

2.3.2.8 REPORTS25_WEBLOC_TRANSLATED

Specifies which directory the report output will be transferred to on the Web server machine if the Reports Server and Web server do not share file systems. This must be the absolute physical path of the virtual directory defined in REPORTS25_WEBLOC in order for there to be no conflict. If the value of this variable would be the same as the cachedir setting in the Reports Server configuration file, you should not specify a value for it.

If this variable is set to an invalid value, R25OWS or R25CGI will not be able to display HTML/PDF output in the client browser ("Cannot access report output" message will result), although the report will run and output will be produced.

See the examples in Section 2.3.2.7, "REPORTS25_WEBLOC".

2.3.2.9 REPORTS25_OWSHHELP

For the Reports Web Cartridge, defines URL/URI of the R25OWS help file, which will be navigated to when R25OWS is invoked with the empty request:

http://your_webserver/r25ows?
For example, setting it to http://www.yahoo.com will go to that URL; setting it to myhelpfile.htm will display the file:

http://your_webserver/myhelpfile.htm

If this parameter is not defined, a default help screen will be displayed.

2.3.2.10 REPORTS25_OWSMAP
For the Reports Web Cartridge, defines fully-qualified file name/location of the R25OWS map file (if map file configuration is used), for example, C:\ORANT\REPORT25\MYMAP.DAT.

2.3.2.11 REPORTS25_OWSDIAGBODYTAGS
For the Reports Web Cartridge, specifies HTML tags which will be inserted as a <BODY…> tag in the R25OWS diagnostic/debugging output. For instance, you may want to use this environment to set up text/background color, image, etc.

2.3.2.12 REPORTS25_OWSDIAGHEADTAGS
For the Reports Web Cartridge, specifies HTML tags which will be inserted between <HEAD> …</HEAD> tags in the R25OWS diagnostic/debugging output. For instance, you may want to use this environment to set up <TITLE> or <META…> tags, etc.

2.3.2.13 REPORTS25_OWSPATHONLYURL
For the Reports Web Cartridge, when defined (the value does not matter), it causes R25OWS to use path-only URLs (for example, “/rscache/...” instead of “http://mywebserv.com:1550/rscache/...”) in HTTP redirection to report output and in the R25OWS-generated diagnostic output. This is useful to force the client browser to retain the original URL request in its “Location:” field after retrieving the report output. Normally, R25OWS will switch the URL to the location of the report output (that is, the directory specified by REPORTS25_WEBLOC), and will show the name of the report output file which, just having been generated, has a rather uninformative name such as 12345679.htm. By setting this parameter, this “URL switching” can be avoided. This will work only if all URLs are using the default port (typically 80).

The drawback to setting this parameter, however, is that any subsidiary files (such as image files) dependent on the main report output file will not be retrieved by the browser (although they are created successfully in the directory pointed to by REPORTS25_WEBLOC). This can be worked around either by specifying a <BASE HREF="..."> or <META HREF="..."> tag (with the main file location as a full or
partial URL) in the report definition so that it is included in the main report output file, or by setting REPORTS25_WEBLOC to point to the location of the R25OWS executable.

Setting REPORTS25_OWSPATHONLYURL can also be used to work around certain Intranet/Internet configurations where the same machine may have several different IP addresses simultaneously (with several network cards), because it removes the dependence of the HTTP redirection on the machine name or IP address.

2.3.2.14 REPORTS25_OWSNODIAG
For the Reports Web Cartridge, when defined, disables all debugging/diagnostic output from R25OWS:

http://your_webserver/r25ows/help?

will not work when REPORTS25_OWSNODIAG is defined.

2.3.2.15 REPORTS25_CGIHLP
For the Reports Web CGI, defines URL/URI of the R25CGI help file, which will be navigated to when R25CGI is invoked with the empty request:

http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe?

For example, setting REPORTS25_CGIHLP to http://www.yahoo.com will go to that URL; setting it to myhelpfile.htm will display the file:

http://your_webserver/myhelpfile.htm

If this variable is not defined, a default help screen will be displayed.

2.3.2.16 REPORTS25_CGIMAP
For the Reports Web CGI, defines fully qualified file name/location of the R25CGI map file if map file configuration is used. For example:

c:\orant\report25\mymap.dat

2.3.2.17 REPORTS25_CGIDIAGBODYTAGS
For the Reports Web CGI, specifies the HTML tags that will be inserted as <BODY...> tags in the R25CGI diagnostic/debugging output. For example, you can use this environment variable to set up text/background color and image.
2.3.2.18 REPORTS25_CGIDIAGHEADTAGS
For the Reports Web CGI, specifies HTML tags which will be inserted between <HEAD> ... </HEAD> tags in the R25CGI diagnostic/debugging output. For example, you may want to use this environment to set up <TITLE> or <META...> tags.

2.3.2.19 REPORTS25_CGINODIAG
For the Reports Web CGI, when defined, disables all debugging/diagnostic output from R25CGI:

http://your_webserver/cgi-bin/r25cgi32.exe/help?

will not work when REPORTS25_CGINODIAG is defined.

2.3.2.20 REPORTS25_CGIPATHONLYURL
For the Reports Web CGI, when defined (the value does not matter), forces R25CGI to return any URLs to the browser as partial, path-only URLs; that is, instead of returning:

Location: http://your_webserver:port/path

It will return only:

Location: /path

This variable can be used to resolve problems with Web server setups that use multiple ports, including a default port.

2.3.2.21 USER_NLS_LANG or NLS_LANG
Defines NLS Language used in your URL request as well as HTML/PDF output (using Oracle NLSRTL Language IDs). For example:

American_America.WE8ISO8859P1

2.3.3 Step 3: Specifying URL run requests
When specifying URL run requests to the Reports Web Cartridge or CGI, you should:

- Make sure all report file names and connection strings are valid on the Reports Server’s file system.
- Specify at least three parameters: SERVER, REPORT, and USERID.
If you are using the Reports Web CGI on Windows, use R25CGI32.EXE (case insensitive). On UNIX, use r25cgim (case sensitive) unless you use a shell script to invoke CGI runtime, in which case you should use the name of the script that calls r25cgim.

The examples below are equivalent to the following command line:

R25CLI SERVER=your_repserver REPORT=breakb.rdf USERID=scott/tiger@MYDB DESTYPE=CACHE DESFORMAT=HTML

For more information, including syntax rules governing URL run requests, refer to the online documents R25OWS.HTM and R25CGI.HTM. For a complete description of the command line parameters and semantics, refer to the online document R25MTSUS.HTM and Reports Runtime online help.

2.3.3.1 A non-URL-encoded (non-standard) request:

Web Cartridge:

http://your_webserver/r25ows?server=your_repserver+report=breakb.rdf+userid=scott/tiger@MYDB+destype=cache+desformat=HTML

CGI:

http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe?server=your_repserver+report=breakb.rdf+userid=scott/tiger@MYDB+destype=cache+desformat=HTML

2.3.3.2 A URL-encoded standard request:

Web Cartridge:

http://your_webserver/r25ows?server%3Dyour_repserver+report%3Dbreakb%2Erdf+userid%3Dscott%2Ftiger%40MYDB+destype%3Dcache+desformat%3DHTML

CGI:

http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe?server%3Dyour_repserver+report%3Dbreakb%2Erdf+userid%3Dscott%2Ftiger%40MYDB+destype%3Dcache+desformat%3DHTML

2.3.3.3 A non-URL-encoded (non-standard) form-style request:
Web Cartridge:

http://your_webserver/r25ows?server=your_repserver&report=breakb.rdf&userid=scott/tiger@MYDB&destype=cache&desformat=HTML

CGI:

http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe?
server=your_repserver&report=breakb.rdf&userid=scott/tiger@MYDB&destype=cache&desformat=HTML

2.3.3.4 A URL-encoded standard form-style request:

Web Cartridge:

http://your_webserver/r25ows?server=your_repserver&report=breakb%2Erdf&userid=scott%2Ftiger%40MYDB&destype=cache&desformat=HTML

CGI:

http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe?
server=your_repserver&report=breakb%2Erdf&userid=scott%2Ftiger%40MYDB&destype=cache&desformat=HTML

2.3.3.5 An HTML form:

Web Cartridge:

<HTML>
<!--Form Action is R25OWS URL-->
<FORM METHOD=POST ACTION="http://your_webserver/r25ows?">
<!--Parameters not exposed to user are hidden-->
<!--INPUT name=foo type=hidden value=bar-->
<CENTER><H1>Set Reports Multi-tier Server Parameters: </H1></CENTER>
<BR>
Report Name:  
<INPUT name=report type=text value="breakb.rdf"><BR>
Database Connection: 
<INPUT name=userid type=text value="scott/tiger@mydb"><BR>
Output Format: 
<SELECT name=desformat>
<OPTION value=HTML selected> HTML
<OPTION value=PDF> PDF
</SELECT> <BR>
CGI:

<!--Form Action is R25CGI URL-->
<FORM METHOD=POST ACTION="http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe?">
<!--Parameters not exposed to user are hidden-->
<INPUT name=foo type=hidden value=bar>
</FORM>

2.3.4 Step 4 (Optional): Using the URL mapping feature

Both the Reports Web Cartridge and Web CGI implement URL mapping. When URL mapping is enabled, the first argument in the URL is treated as a special key. This key maps to command line parameter-value combinations in a key mapping file set up by a Web site administrator.

URL Mapping is useful for:

- Shortening the URL, making it more convenient to use.
- Remapping the URL run configuration without having to change the original URL.
- Standardizing several typical run configurations for the organization.
- Hiding certain parameters from the end user (for example, database connection string).
- Restricting end user choice as to which parameters may be used for the request.

For more information on how to implement URL mapping, refer to the online documents R25OWS.HTM and R25CGI.HTM.
2.3.5 Step 5: Deploying reports on the Web

You create the Reports Server in the same way you do for any other environment, using the Report Wizard and the Report Editor. You can then add Web functionality to the report using the Web Wizard and the Report Editor. The Web Wizard enables you to quickly add basic Web functionality (for example, bookmarks). The Report Editor enables you to add more advanced functionality (for example, custom Report Escapes and URLs).

After the reports you want to deploy on the Web are ready, copy the .RDF files to the appropriate directory on the application server file system. Be sure to generate the .RDF files on the same platform as the application server on which you will deploy them.

You can deploy .RDF files in any directory on your application server, provided you have defined the cartridge parameters and environment variables correctly. Refer to Section 2.3.2.4, “Cartridge Parameters and Environment Variables”.

Once you have copied .RDF files to the application server, you must link them to a Web page and notify end users of their availability. For more information, refer to Section 2.1.3.6, “Step 3c: Providing URLs of your Web-enabled applications”.
### 2.4 Troubleshooting your Reports Server configuration

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports are not running.</td>
<td>Make sure the Web server is responding. Type:</td>
</tr>
<tr>
<td></td>
<td><code>http://your_webserver</code></td>
</tr>
<tr>
<td></td>
<td>in your browser URL window. A Web server administration page should appear; if not, see your Web server installation documentation.</td>
</tr>
<tr>
<td></td>
<td>Make sure your Web Cartridge or CGI executable has been found and is responding. For Win 32, type one of the following:</td>
</tr>
<tr>
<td></td>
<td><code>http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe</code></td>
</tr>
<tr>
<td></td>
<td><code>http://your_webserver/r25ows</code></td>
</tr>
<tr>
<td></td>
<td>For UNIX, type:</td>
</tr>
<tr>
<td></td>
<td><code>http://your_webserver/your_virtual_cgi_dir/r25cgim</code></td>
</tr>
<tr>
<td></td>
<td>in your browser URL field.</td>
</tr>
<tr>
<td></td>
<td>A help page should appear. If it does not, check the definition of your_virtual_bin_directory (usually called cgi-bin) in your Web server configuration file. It should be mapped to an existing physical directory on your Web server. You must have a copy of the R25CGI executable in this physical directory.</td>
</tr>
<tr>
<td></td>
<td>Make sure that the environment variable REPORTS25_OWSNODIAG or REPORTS25_CGINODIAG is not defined, otherwise all diagnostic output will be disabled. Test this by typing one of the following:</td>
</tr>
<tr>
<td></td>
<td><code>http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe/showenv?</code></td>
</tr>
<tr>
<td></td>
<td><code>http://your_webserver/r25ows/showenv?</code></td>
</tr>
<tr>
<td></td>
<td>This will also allow you to view the other cartridge parameters or environment variables.</td>
</tr>
<tr>
<td></td>
<td>Check the list of cartridge parameters or environment variables and make sure that REPORTS25_WEBLOC and REPORTS25_WEBLOC_TRANSLATED both are defined. The former is defined to be a virtual directory on the Web server; the latter is defined to be the absolute path of the directory which the virtual directory maps to.</td>
</tr>
</tbody>
</table>
Troubleshooting your Reports Server configuration

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
</table>
| Reports are not running.    | Make sure the environment variable REPORTS25_PATH is defined. If you do not plan to use URL mapping, make sure that REPORTS25_OWSMAP or REPORTS25_CGIMAP is undefined. Check to be sure that there is no file named owscmd.dat or cgicmd.dat in the ORACLE_HOME\REPORT25\ directory. Confirm that a map file is not found by typing one of the following:  
  http://your_webserver/you_virtual_cgi_dir/r25cgi32.exe/showmap?  
  http://your_webserver/r25ows/showmap?  
  Try running a simple report to your browser, by typing one of the following:  
  http://your_webserver/your_virtual_cgi_dir/  
  r25cgi32.exe?report=your_report.rdf+userid=scott/tiger@mydb+desformat=html  
  http://your_webserver/r25ows?server=your_repserver+report=your_report.rdf+userid=scott/tiger@mydb+desformat=html  
  If the report does not display, check to be sure that:  
  ■ Your_report.rdf runs correctly from Reports or Reports Runtime.  
  ■ Your_report.rdf is located in a directory specified under REPORTS25_PATH.  
  ■ The database connection string is correct.  |
| URL mapping is not working. | Make sure you have a valid key mapping file. It must either be named owscmd.dat or cgicmd.dat in the S25 directory, or named according to the value of the environment variable REPORTS25_OWSMAP or REPORTS25_CGIMAP.  
  Test running a report by typing one of the following:  
  http://your_webserver/your_virtual_cgi_dir/r25cgi32.exe?yourkey+report=your_report.rdf+userid=scott/tiger@mydb+desformat=html  
  http://your_webserver/r25ows?yourkey+server=your_repserver+report=your_report.rdf+userid=scott/tiger@mydb+desformat=html  
  where your key is a valid key entry in the key mapping file. |
2.5 Configuring the Graphics Server

To deploy Graphics displays on the Web, refer to the following instructions:

- Section 2.5.1, “Creating virtual directories on your Web server”
- Section 2.5.2, “Creating the Graphics Server cartridge”
- Section 2.5.3, “Making the Graphics web cartridge available to end users”

2.5.1 Creating virtual directories on your Web server

- To create pointers to various physical directories on your application server, create virtual directories on your Web server.

  For more information about creating virtual directories, refer to your Web Server documentation.

2.5.2 Creating the Graphics Server cartridge

Note: The following instructions are specific to Oracle Web Application Server 3.0. If you are using Oracle WebServer 2.1, refer to Section C.3, “Creating the Graphics Server Cartridge Entry in the Web Request Broker”. If you are using another Web server, refer to your Web server documentation for information about creating application cartridges.

1. Install Oracle Web Application Server 3.0, if you haven’t already.

   Note: When you install the Web Application Server, accept WWW as the name of the default Web Listener. You specify the listener name in the Oracle Web Listener Configuration dialog.

2. (UNIX only) Set the GRAPHICS_WEB_DIR and OWS_IMG_DIR environment variables.

---

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error reported when opening the report.</td>
<td>Check the name and extension carefully. On UNIX machines, the actual report name must be in the same case as specified in the URL. If you are using Windows Explorer in Windows, be sure not to hide extensions for the displayed files that you are copying and renaming. (Check View—&gt;Options in the Explorer window.) This prevents you from creating files with names like &quot;your_report.rdf.txt.&quot; Alternatively, use a DOS window for file manipulation.</td>
</tr>
</tbody>
</table>
Configuring the Graphics Server

- Set `GRAPHICS_WEB_DIR` to the full directory path that contains the .OGDs that you want to run on the Web.
- Set `OWS_IMG_DIR` to the location of the `/ows-img/` virtual directory. You define the `/ows-img/` virtual directory in the Oracle Web Application Server Administration page.

**Note:** On UNIX, you must start the listener from the shell where you set the `GRAPHICS_WEB_DIR` and `OWS_IMG_DIR` environment variables.

**Note:** On Windows NT, the Oracle Installer will set `GRAPHICS_WEB_DIR` to `ORACLE_HOME\GRAPHICS25\WEB_OGD`, and set `OWS_IMG_DIR` to `ORACLE_HOME\OWS\3.0\ING`. You can verify or change these values using `regedit`.

3. Start the Web Request Broker listener. For example, type:
   ```
   owsctl start wrb
   ```
   For more information about the listener, refer to your Oracle WebServer documentation.

4. Start the Admin listener. For example, type:
   ```
   owsctl start admin
   ```

5. Start the WWW listener. For example, type:
   ```
   owsctl start www
   ```

6. Start your browser.

7. Access the Oracle Web Application Server by navigating to the appropriate URL (`http://<name of your machine or IP address>:<port number of Web Server Admin listener>`).

   For example: `http://myserver.com:8888`.
   The Oracle Web Application Server appears.

8. Click **Web Applications Server Manager** to display the Administration page.
9. Click **Oracle Web Application Server**.
10. Click **Cartridge Administration** to display the Cartridge Administration page.
11. Click **Add New Cartridge**.
12. Click **Add New Cartridge with Manual Configuration**, then enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this value...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartridge Name</td>
<td>The cartridge name. For example: OGWEB</td>
</tr>
<tr>
<td>Object Path</td>
<td>The directory path and file name of the Forms Cartridge Handler.</td>
</tr>
<tr>
<td></td>
<td>- On Microsoft Windows NT: <code>%ORACLE_HOME%\bin\gw25w32.dll</code></td>
</tr>
<tr>
<td></td>
<td>- On UNIX: <code>$ORACLE_HOME/lib/libgw25.so</code></td>
</tr>
<tr>
<td>Entry Point</td>
<td>GWWRBMain</td>
</tr>
<tr>
<td></td>
<td>GWWRBMain is the name of the entry point routine in the cartridge shared library.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> GWWRBMain is the required value for Entry Point.</td>
</tr>
<tr>
<td>Minimum # of Instances</td>
<td>The minimum number of users that can simultaneously connect to the cartridge. For example: 0</td>
</tr>
<tr>
<td>Maximum # of Instances</td>
<td>The maximum number of users that can simultaneously connect to the cartridge. For example: 100</td>
</tr>
<tr>
<td>Virtual Path (under Virtual Paths)</td>
<td>At runtime, end users pass URLs (HTTP requests) to your Web server. If a request contains the virtual directory associated with your cartridge, the Web server automatically transfers the request to the Web Request Broker. The Web Request Broker in turn starts the cartridge associated with the virtual directory. When defining the Virtual Path, do not include a trailing slash. For example: /ogweb</td>
</tr>
<tr>
<td>Physical Path (under Virtual Paths)</td>
<td>The directory path for the Forms Cartridge Handler:</td>
</tr>
<tr>
<td></td>
<td>- On Microsoft Windows NT: <code>%ORACLE_HOME%\bin\</code></td>
</tr>
<tr>
<td></td>
<td>- On UNIX: <code>$ORACLE_HOME/bin/</code></td>
</tr>
</tbody>
</table>
13. Click **Register New Cartridge**.

   The Web Application Server registers your cartridge and creates a link for your cartridge in the Web Application Server Home page.

14. Click your cartridge link to display the Cartridge Configuration page.

15. Click **Cartridge specific parameters**, then enter the following parameters and parameter values:

   **Note:** Cartridge parameters and parameter values are case-sensitive.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GW_TIMEOUT</td>
<td>The number indicating the time interval (in minutes) for which the display will remain open. When the time interval specified here has elapsed, the display is automatically closed. For example: 30</td>
</tr>
</tbody>
</table>

16. (Optional) Enter the following optional cartridge parameters and parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GW_IMAGES_USE_FILES</td>
<td>The value of this parameter determines whether the Graphics Server creates temporary GIF image files on the server (in the web_tmp directory in the img directory in the web server path) when returning HTML pages that show Graphics displays. If the value is NO, then the HTML page instead references the Graphics Server (a second time) to retrieve the image data, which is “streamed” back to the client by the server as binary content without creating a temporary file on the server. The default value of this parameter (if it is omitted entirely) is YES. <strong>Note:</strong> Creating temporary GIF files can introduce an administrative task of periodically removing these files from the server (once you are certain they are not being used). However, the Graphics cartridge automatically removes old image files. Using temporary files can result in some “unused” image files on the server; however, this is offset by a small performance benefit, since using “streamed” images is slightly slower. Using “streamed” images is slower because image requests must also be routed through the Web Request Broker to the Graphics Server (instead of ordinary requests to the server for image files which are optimized operations).</td>
</tr>
</tbody>
</table>
17. Click **Modify Cartridge Configuration**.

18. To activate your changes, do the following:
   - Click **Listener**.
   - Click **Stop** to stop the WWW listener.
   - Click **Start** to restart the WWW listener.

### GW_LINKS_CLOSE_PREV

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GW_LINKS_CLOSE_PREV</td>
<td>Determines whether the Graphics Server closes the current OGD file when navigating to another OGD file via an embedded hyperlink (that is, URL). If the parameter value is YES, and an embedded URL is used to open another OGD on any Graphics Server (that is, if it contains the string “openfile”), then the current OGD will be closed and the new one opened. This conserves resources on the server, since the user no longer needs the first OGD to remain open. <strong>Note:</strong> Other types of URLs (for example, to another web-page or HTML object) will operate as usual, and the OGD containing these links will remain open after executing them, regardless of this parameter value. If the user returns to the first OGD, either via the “back” button on their browser or via another URL, they will see a “fresh” version of this document; any changes they made before will not appear. The default value of this parameter is YES. This parameter value may be overridden by setting the bind-variable <code>OG_URL_CLOSE_THIS_DOC</code> to YES or NO in the OGD PL/SQL function that invokes the embedded URL (that is, via <code>OG_URL</code> and <code>OG_URL_TARGET</code>). <strong>Note:</strong> The value of <code>OG_URL_CLOSE_THIS_DOC</code> must be explicitly set each time (before a link is executed) to override this parameter value, because it is only meant for special cases.</td>
</tr>
</tbody>
</table>

### GW_WRITE_TRACE_FILE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GW_WRITE_TRACE_FILE</td>
<td>The value of this parameter determines whether the Graphics Server writes a “trace” file to the server's filesystem. The trace file can be used by an administrator for debugging purposes. The name of this file is <code>gwtrace.txt</code>. Because trace files are required in special situations, the default value of this parameter is NO.</td>
</tr>
</tbody>
</table>
19. (UNIX only) Before running the cartridge, create the directory
$OWS_IMG_DIR/web_tmp, then give it global write permissions.

For example, type:

```
mkdir $OWS_IMG_DIR/web_tmp
chmod 777 $OWS_IMG_DIR/web_tmp
```

**Note:** On Windows NT, the Oracle Installer will create this directory automatically.

### 2.5.3 Making the Graphics web cartridge available to end users

Once you have created displays that you want to deploy on the Web, you must provide a method for users to access them. To do so:

- Place your organization’s displays in the directory set by the GRAPHICS_WEB_DIR environment variable (the default is ORACLE_HOME\GRAPHICS25\WEB_OGD). This makes them available to users through the pull-down list on the Graphics Web toolbar.

- Inform users of the URL that accesses the cartridge you created. The URL is in the format:

```
http://your_server/your_virtual_path
```

The end point of the URL corresponds to the cartridge value you specified in the Virtual Path field (when you created the cartridge entry in the Web Request Broker). For example, if you specified “/ogweb” as the Virtual Path, the URL would be:

```
http://your_server/ogweb
```
This section covers these topics:

- General guidelines for designing Web applications
- Guidelines for designing Forms Web applications
- Guidelines for designing Reports Web applications
- Guidelines for designing Graphics web applications

3.1 General guidelines for designing Web applications

- Seriously consider network factors that affect the performance of your Web applications (for example, interaction with security firewalls, heavy user loads, frequent network roundtrips to application and database servers, and other factors).

- Limit the number of image items and background images you include in your forms and reports. Each time an image is required, it must download from the application server. To display a company logo, for example, try including the image in the HTML page that downloads at application startup instead of retrieving it from the database or filesystem. For information customizing the startup HTML page, see Section 3.1.2, “Creating your own template HTML files” at the end of this chapter.

- Optimize your network connections where possible. Refer to your SQL*Net documentation for hints on enhancing SQL*Net performance.

- Design your queries to execute as efficiently as possible, and ensure PL/SQL program units are compiled. For more information, refer to the Oracle7 Server Concepts Manual and the PL/SQL User’s Guide and Reference.
3.1.1 Using RUN_PRODUCT to integrate Developer/2000 applications

To invoke Reports or Graphics from a Web-enabled form, use the Forms built-in subprogram RUN_PRODUCT.

3.1.1.1 Running Reports Reports from a Web-enabled Forms application

To use RUN_PRODUCT to run a report from a form running on the Web, you must set three environment variables:

Table 3-1 Environment variables for using RUN_PRODUCT in web applications

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMS45_OUTPUT</td>
<td>Physical directory on the application server in which to store generated Reports files. For example: c:\orant\forms45\my_reps\</td>
</tr>
<tr>
<td>FORMS45_MAPPING</td>
<td>Virtual directory pointing to the physical directory defined by the FORMS45_OUTPUT variable. For example: /web_reps/</td>
</tr>
<tr>
<td>FORMS45_REPFORMAT</td>
<td>Format in which to store generated Reports output. For example: PDF or HTML</td>
</tr>
</tbody>
</table>

**Note:** On Windows NT, you define your environment variables in the Registry. On UNIX, you define your environment variables in the command shell. For more information on setting up environment variables, refer to the Developer/2000 installation instructions for your operating system.

After you set the environment variables above, the following sequence occurs automatically when a form running on the Web calls RUN_PRODUCT to invoke Reports:

If the output format of the report is FILE:

- The report does not display in the user’s browser.
- The resulting file is stored in the physical directory specified by the FORMS45_OUTPUT environment variable.
- The file name of the report file is the same name that is defined in the form definition.

---

3-2  Developer/2000: Deploying Applications on the Web
If the output format of the report is `SCREEN` or `PREVIEW`:

- The resulting output is stored (as a temporary file with an auto-generated file name) in the physical directory specified by the `FORMS45_OUTPUT` environment variable.
- The Web server looks for the temporary file name (in the virtual directory defined by the `FORMS45_MAPPING` environment variable).
- The Web server checks the desired display format specified by the `FORMS45_REPFORMAT` environment variable, and displays the report in that format in the end user’s browser.

**Note:** For detailed information on `RUN_PRODUCT` and its syntax, refer to the “RUN_PRODUCT” topic in the Forms online help.

### 3.1.1.2 Running Graphics from a Web-enabled Forms application

If your Web-enabled forms call `RUN_PRODUCT` to show a Graphics display, you do not have to set any special environment variables.

### 3.1.2 Creating your own template HTML files

Consider creating your own static and cartridge HTML file templates (by modifying the templates provided by Oracle). By doing this, you can hardcode standard Forms Client applet 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.

For more information, see Section A.3, “Working with the Static (Non-Cartridge) HTML File Template” and Section A.2, “Working with the Cartridge HTML File Template”.

### 3.1.3 Creating an HTML application menu

As you deploy more and more Forms and Reports applications on the Web, try creating a single HTML page to serve as a centralized menu for your various Web-enabled applications. This approach eliminates the need to broadcast the URL of every application you deploy or remove. As you change your roster of available applications, simply modify the collection of links on the Web menu. End users then contact the menu URL and select from the list of available forms and reports.
Figure 3–1  Web Application Menu
3.2 Guidelines for designing Forms Web applications

When designing Forms applications for the Web, consider the guidelines discussed in:

- Section 3.2.1, “Using Designer/2000 with the Developer/2000 Server”
- Section 3.2.2, “Reducing network traffic”
- Section 3.2.3, “Creating a generic GUI for Web-enabled Forms applications”
- Section 3.2.4, “Selecting fonts”
- Section 3.2.5, “Using the SHOW_DOCUMENT built-in”
- Section 3.2.6, “Working with images”
- Section 3.2.7, “Feature restrictions for forms on the Web”

3.2.1 Using Designer/2000 with the Developer/2000 Server

The Developer/2000 Server supports forms generated by Designer/2000 (32-bit, Release 1.3.2 or higher). If you use the standard Designer/2000 forms generator templates (ofg4pc1t.fmb and ofg4pc2t.fmb) to generate form and menu definitions, you can use the Developer/2000 Server to compile .FMX and .MMX files and immediately run the applications on the Web.

3.2.2 Reducing network traffic

To cut down on the number of network roundtrips required for users to operate your Forms applications on the Web, consider reducing or eliminating the following Forms features in your applications:

- **Mouse triggers.** Including When-Mouse-Click, When-Mouse-DoubleClick, When-Mouse-Down, and When-Mouse-Up triggers in your forms will impact speed and performance; the Forms Client must communicate with the Forms Server (necessitating a network roundtrip) each time one of these trigger fires. The When-Mouse-Move trigger is not supported due to the high number of network roundtrips required each time it fires.

- **Timers.** If your form includes a timer that fires every 1/100th of a second, end users face the performance ramifications of 60,000 network roundtrips every minute. Either reduce the number of timers in your forms, or change the timing interval on which your timers fire.

- **Image items and background images.** See Section 3.2, “Guidelines for designing Forms Web applications”.

Guidelines for Designing Web Applications 3-5
3.2.3 Creating a generic GUI for Web-enabled Forms applications

Any end user with a Java-enabled Web browser can run your Web-enabled Forms applications. To make sure the application’s appearance does not vary from end user to end user—from Web browser to Web browser—you must create a GUI that is generic across platforms.

3.2.4 Selecting fonts

Most fonts are not supported across all platforms. For example, San Serif is a commonly-used font in Microsoft Windows applications. However, San Serif is not available in UNIX. When a font is not available on a platform, Forms attempts to use a similar font. As a result, when designing forms to deploy on the Web, be sure to follow the font guidelines below.

At runtime, the Forms Server maps a form’s fonts into their Java equivalents. Java then renders the font in a font pre-defined for the deployment platform. To convert your form’s fonts into Java equivalents, Java uses an alias list, located in the file called Registry.dat.

The following table lists the Java fonts, and their equivalents on the major deployment platforms:

<table>
<thead>
<tr>
<th>Java Font</th>
<th>Windows Font</th>
<th>X Windows Font</th>
<th>Macintosh Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier</td>
<td>Courier New</td>
<td>adobe-courier</td>
<td>Courier</td>
</tr>
<tr>
<td>Dialog</td>
<td>MS Sans Serif</td>
<td>b&amp;h-lucida</td>
<td>Geneva</td>
</tr>
<tr>
<td>DialogInput</td>
<td>MS Sans Serif</td>
<td>b&amp;h-lucidatypewriter</td>
<td>Geneva</td>
</tr>
<tr>
<td>Helvetica</td>
<td>Arial</td>
<td>adobe-helvetica</td>
<td>Helvetica</td>
</tr>
<tr>
<td>Symbol</td>
<td>WingDings</td>
<td>itc-zapfdingbats</td>
<td>Symbol</td>
</tr>
<tr>
<td>TimesRoman</td>
<td>Times New Roman</td>
<td>adobe-times</td>
<td>Times Roman</td>
</tr>
</tbody>
</table>

**Note:** If a font from your form does not map to a Java font (through the Forms font alias table), Java automatically assigns a Java font to the unmapped application font. To change the Java font alias scheme, simply edit Registry.dat.

3.2.5 Using the SHOW_DOCUMENT built-in

You can display a Web page from within a Web-deployed Forms application via the SHOW_DOCUMENT built-in. This built-in enables you to specify the URL of a Web page, and where you want to display the Web page.
3.2.6 Working with 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 end 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 (instead of including it as a background image in the application, where it must be retrieved from the database or filesystem and downloaded repeatedly to users’ machines). For information on customizing the startup HTML page, see Section 3.1.2, “Creating your own template HTML files”.

3.2.7 Feature restrictions for forms on the Web

When designing forms for eventual deployment on the Web, keep in mind that certain Forms features behave differently—or not at all—when a form is deployed on the Web. The following table lists Forms features, whether the feature is supported on the Web, and any notes or guidelines about the feature:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported?</th>
<th>Guidelines and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveX, OCX, OLE, VBX</td>
<td>No</td>
<td>Third-party controls that display screen output on the application server are not supported because end users cannot view the output.</td>
</tr>
<tr>
<td>combo boxes</td>
<td>No</td>
<td>Available in Developer/2000 Release 2.1, pending Java implementation of combination box widget.</td>
</tr>
<tr>
<td>console</td>
<td>Yes</td>
<td>To display the console (includes the status and message lines) to end users, set the form-level property Console Window to the window in which you wish to display the console.</td>
</tr>
<tr>
<td>firewall</td>
<td>Yes</td>
<td>To deploy Forms applications on the Internet, you must run the Forms Server outside a security firewall (if any), and you must have a tunnel (through the firewall) that supports SQL*Net.</td>
</tr>
</tbody>
</table>
3.3 Guidelines for designing Reports Web applications

This section covers the following topics:

- Section 3.3.1, “About Web reports”
- Section 3.3.2, “Adding Web functionality”
- Section 3.3.3, “Selecting the output format”
- Section 3.3.4, “Helpful tips”

3.3.1 About Web reports

Reports enables you to generate files for display on the Web in the following formats:

- HTML can be displayed in an HTML 3.0-compliant Web browser, such as Netscape version 2.2 or higher.
- HTML with cascading style sheets can be displayed by an HTML 3.0-compliant Web browser that supports cascading style sheets (for example, Microsoft Internet Explorer 3.01).
- PDF can be displayed in a standalone or plug-in PDF viewer, such as Adobe Acrobat Reader.
While it is possible to generate HTML or PDF output without changing your report definition, the report is often more useful on the Web if you add Web functionality (for example, bookmarks and hypertext links) to it. The Web Wizard in Reports is the quickest way to prepare a report for the Web. It enables you to add basic Web functionality and generate HTML or PDF output immediately. If you want to add further Web functionality or tweak what the Web Wizard has done, you can use the Report Editor and/or the Object Navigator.

**Note:** Web functionality in a report is ignored when you are not generating HTML or PDF. Adding Web functionality will not interfere with your ability to run the report to more traditional destination types (printers, ASCII files, etc.).

### 3.3.2 Adding Web functionality

To make your report more useful on the Web, Reports provides the capability to add Web functionality to it. You can add Web functionality to a report in any of the following ways:

- **Web Wizard** In the Web Wizard, you can specify bookmarks and HTML files to insert at the beginning and end of the report.

- **Property Palette** In the Property Palette for the report, you can enter Report Escapes that define HTML to be inserted at the beginning and end of the report, and at the beginning and end of pages. In the Property Palette for layout objects, you can specify Web Link properties that make the object a hyperlink destination or hyperlink, or execute a PDF application command line.

- **SRW.SET_ATTR** `SRW.SET_ATTR` enables you to set Web properties programatically in triggers. For example, you could use `SRW.SET_ATTR` in a format trigger for a layout object to make it a hyperlink or hyperlink destination.

#### 3.3.2.1 Hyperlinks

Hyperlinks (or hypertext links) allow users to navigate to different sections of a report or other documents on the Web. To create a hyperlink, you first create a target (or hyperlink destination), which is a unique identifier for the current location. Next, you create a hyperlink elsewhere in the same or a different document that points to the target. When users click on the hyperlink, they will be taken to the target to which it points.

A hyperlink can take you to:

- Another document on the same machine. For example:

  file:/private/mynewdoc.pdf or file:///C:/temp/mynewdoc.pdf
Another document on a different machine. For example:
http://www.newmach.com/newdoc.pdf

A destination within the current document. For example:
#my_dest_name

A destination within a local document. For example:
file:/private/somedoc.pdf#a_dest_name

A destination within a remote document. For example:
http://www.newmach.com/newdoc.pdf#some_dest_name

Any URL. For example:
http://www.newmach.com/newdoc.html
ftp://www.reposit.com/filetoget.example
http://www.somemch.com/cgi-bin/webmenu?choice1

In the example below, a top-level report contains hypertext links that display more detailed reports for each region. A hypertext contact address displays an e-mail template for sending comments to the Webmaster.
Figure 3–2 Example of a hypertext report

Example 3–1 Hyperlink PL/SQL example

The format trigger below defines a hypertext link to a destination (mytarget) in another document (newdoc.pdf) for the boilerplate object B_2.

Note: If the target were in the same document, you would omit http://www.newmach.com/newdoc.pdf.

```plsql
function B_2FormatTrigger return boolean is
    begin
        srw.attr.mask := srw.HYPERLINK_ATTR;
        srw.attr.hyperlink :=
            'http://www.newmach.com/newdoc.pdf' ||
            '#mytarget';
        srw.set_attr(0,srw.attr);
    end;
```

Guidelines for Designing Web Applications  3-11
Example 3–2 Dynamic hyperlink example

The format trigger below defines a hypertext link for the boilerplate object $B_3$. Notice how the destination of the link is determined dynamically based upon the values of $SRCDIR$ and $CF\_PICKVIDEO$. For example, if the value of $SRCDIR$ is http://www.newmach.com, and the value of $CF\_PICKVIDEO$ is good.avi, this function would assign the following hypertext link to the object:

http://www.newmach.com/webdemo/src/good.avi

Format trigger:

```javascript
function B_3FormatTrigger return boolean is
begin
srw.attr.mask := srw.HYPERLINK_ATTR;

srw.attr.hyperlink := :srcdir||:cf_pickvideo;
if ( upper(:cf_pickvideo) like '%GOOD%' ) then
  srw.attr.mask := srw.attr.mask +
  srw.FBCOLOR_ATTR +
  srw.BBCOLOR_ATTR;
  srw.attr.fbgcolor := 'green';
  srw.attr.bbgcolor := 'green';
end if;
srw.set_attr(0,srw.attr);
return (TRUE);
end;
/* SRCDIR is a parameter whose value is determined at
** runtime by the following After Parameter Form Trigger
*/
function AfterPForm return boolean is
begin
  :srcdir := :web_server || '/webdemo/src/';
  :docsdir := :web_server || '/webdemo/docs/';
  return (TRUE);
end;
/* CF\_PICKVIDEO is a formula column whose value is
** determined by the following function
*/
function CF\_pickvideoFormula return Char is
begin
  if ( :avg_h_div < .80 )
    then return ('bad.avi');
  else return ('good.avi');
```

3-12 Developer/2000: Deploying Applications on the Web
3.3.2.2 Bookmark

A bookmark is a string that will appear in a frame of the master HTML document or in the PDF viewer if you open the bookmark area. Clicking on the bookmark displays the associated object at the top of the window. A list of bookmarks can serve as an outline or Table of Contents for your report.

For reports with master-detail relationships, the Web Wizard can create bookmarks for you automatically. You can also create bookmarks by specifying the Bookmark property via the Property Palette or `SRW.SET_ATTR` in a format trigger. In the example below, a PDF report contains bookmarks that correspond to different sections of the report.

**Figure 3–3 Example of a bookmark report**

By default, the order in which bookmarks appear in the list is determined by when Reports renders the objects when you generate the report. You can hierarchically order bookmarks by embedding ordering information in the bookmark definition string. A string with explicit ordering information takes the form:

```
x#book_mark_name
```

where `x` is an outline number. The pound sign (#) and outline number do not appear in the bookmark area but are used to determine order and indentation. For example:

```
1#Expense Summary Section
2#Expense Detail Section
2.1#Expenses for the Administration Department
2.2#Expenses for the Engineering Department
2.3#Expenses for the Sales Department
2.3.1#Expenses for the Eastern Sales Region
2.3.2#Expenses for the Northern Sales Region
2.3.3#Expenses for the Southern Sales Region
2.3.4#Expenses for the Western Sales Region
```

**Example 3–3 Example of bookmark with ordering information**

The format trigger below defines a bookmark for the boilerplate object B_2. Note the use of explicit ordering information (1#) in this example. If you did not want the ordering information, you could omit 1#.

```javascript
function B_2FormatTrigger return boolean is
begin
end;
end if;
end;
```
Guidelines for designing Reports Web applications

Example 3–4  Example of dynamic bookmark

The format trigger below defines a bookmark for the boilerplate object B_2. Notice that the name of the bookmark is dynamic. CATEGORY is a column value that is concatenated with the string Expense Summary Section for each execution of the Format Trigger. In this case, CATEGORY could contain ordering information (for example, 1#) or perhaps a string that makes the bookmark unique within the report.

```
function B_2FormatTrigger return boolean is
begin
  srw.attr.mask := srw.BOOKMARK_ATTR;
  srw.attr.bookmark  := :category || ' Expense Summary Section';
  srw.set_attr(0,srw.attr);
  return (TRUE);
end;
```

3.3.2.3 Action

(PDF only) An action is a command line (for example, `c:\orawin\bin\r25run userid=scott/tiger report=example.rdf` or `/usr/local/bin/phone smith`) that will be executed on the local machine when the object is clicked in the PDF viewer.

Note: An object that is associated with an action cannot also be the source of a hypertext link.

3.3.2.4 Report Escapes

(HTML Only) Report Escapes enable you to insert HTML code at certain points in your report output. This feature is useful for such things as adding consistent headers/footers to each page or giving all of your HTML reports a consistent look and feel. You can enter the HTML code you want to insert directly or as a file.

Report Escapes can be entered in the Property Palette for the report or using `SRW.SET_ATTR` packaged procedure. The Report Escapes are listed below with their `SRW.SET_ATTR` counterparts in parentheses:

- **Before Report** (`BEFOREREPORT_ESCAPE`) The Before Report escape inserts the specified text, graphics, or HTML commands at the beginning of your
document. This attribute is useful for placing a logo or some standard links at the beginning of an HTML document.

- **After Report** *(AFTREPORT_ESCAPE)* The After Report escape inserts the specified text, graphics, or HTML commands at the end of your document. This attribute is useful for placing a logo or some standard links at the end of an HTML document.

- **Before Page** *(BEFPAGE_ESCAPE)* The Before Page escape inserts the specified text, graphics, or HTML commands at the beginning of pages of your document. This attribute is useful for placing a logo or some standard links at the beginning of each page in an HTML document.

- **After Page** *(AFTPAGE_ESCAPE)* The After Page escape inserts the specified text, graphics, or HTML commands at the end of pages of your document. This attribute is useful for placing a logo or some standard links at the end of each page in an HTML document.

- **Before Form** *(BEFFORM_ESCAPE)* The Before Form escape inserts the specified text, graphics, or HTML commands at the beginning of your runtime parameter form.

- **After Form** *(AFTFORM_ESCAPE)* The After Form escape inserts the specified text, graphics, or HTML commands at the end of your runtime parameter form.

For more information on how to add Web attributes to reports, refer to the Report Builder online help.

### 3.3.3 Selecting the output format

There are several differences between HTML and PDF output. Understanding the differences will help you to choose the best format for your Web reports.
3.3.3.1 About HTML output

In general, HTML reports display faster because they are in ASCII format. Here is an example of an HTML report:

![HTML report example](image_url)

Figure 3–4 Example of an HTML report
The fidelity of HTML output to your printed output is not as high as with PDF format. Keep in mind the following:

**Backgrounds**

Background (fill) and border (line) colors/patterns for text are not available in HTML. Bold, italic, underline, and foreground (text) color are supported if the browser supports them.

**Bookmarks**

Bookmarks cause multiple HTML files to be created. One master file is created with two frames, one for bookmarks and one for the report output. One HTML file is created for each of these frames. The master document file name is the name specified in the DESNAME parameter. The bookmark file name is `<desname>b.htm`. The report output file name is `<desname>d.htm`.

**Buttons**

Buttons with text labels are converted to text. Buttons with icons are converted to GIF files. In either case, the format trigger for the button is executed (for example, you could create a hypertext link for the button in the format trigger using `SRW_SET_ATTR`). The action associated with the button (such as play multimedia or execute PL/SQL) is not preserved. For example, if a button is supposed to play a multimedia file, the video is not played when you click the button in the HTML output. The only way to have the button be “active” in HTML output is to use `SRW_SET_ATTR` attributes in its format trigger.

**Drawings**

The only drawn object supported in HTML is a solid, black, horizontal line. The line width specified in the report may be honored depending upon the browser. All other drawn objects (for example, rectangles or circles) in the report layout will not show up in the HTML output. Space for these drawn objects is reserved, but there is no visible representation in the HTML output.

**Fonts**

HTML has seven sizes for text. However, the user can override the size specified in the HTML file from their browser. If the font used in the report is non-proportional, TeleType mode is turned on for the generated HTML file.

**Frames**

Report frames are not visually represented in the HTML output. Any fill or border attributes of frames do not appear in HTML output.
Images

Linked images, image fields, and charts in a report cause GIF files to be created and referenced from the HTML document. Note that even if the linked boilerplate or image field refers to an external GIF file, a new GIF file is generated. If an image is stored in the database, one GIF file may be generated for each occurrence of the image in the report. If an image is stored in a file (for example, imported images, linked boilerplate, or image fields that reference files), only one GIF file will be generated per image regardless of how many times it is repeated in the report. Any GIF image files generated for HTML output have a number sequence (for example, <desname>0.gif ... <desname>17.gif).

OLE objects

When generating HTML output, OLE objects are ignored, but space is reserved in the output for the objects.

Overlapping objects

Objects cannot overlap one another. For example, you could not have text on top of an image. If objects overlap slightly (two characters or less), then the object underneath is truncated to prevent overlap. If objects overlap significantly (or one is completely on top of the other), then the underneath object is removed altogether. In this case, any linking information of the removed object is transferred using the same rules as if it were a frame. Text always takes precedence over horizontal lines, regardless of which is on top. This prevents the line underneath a column label from eliminating the label text.

Pagination

HTML does not have the concept of a page. A separator line is placed between each page of the report. Depending upon the browser, you may or may not need to scroll to see the entire report page. Furthermore, if you print the HTML document from your browser, the printer will not necessarily print the separator lines at the bottom of each page. If you do not want the separator line or you want to use a different separator line, you can use the After Page escape to change it.
3.3.3.2 About PDF format

In general, PDF output results in high-fidelity reports and the convenience of a single output file. Here is an example of a PDF report:

![Figure 3–5 Example of a PDF report](http://webdemo/out/trades.pdf)

PDF reports can take longer to display and are less portable, requiring a PDF viewer or plug-in to view. Here are some additional differences:

**Buttons**

Buttons with text labels are converted to text. Buttons with icons are converted to GIF files. In either case, the format trigger for the button is executed (for example, you could create a hypertext link for the button in the format trigger using `SRW.SET_ATTR`). The action associated with the button (such as play multimedia...
or execute PL/SQL) is not preserved in the PDF output. For example, if a button is supposed to play a video, the video will not be played when you click the button in PDF output. The only way to have the button be “active” in PDF output is to use SRW.SET_ATTR attributes in its format trigger.

**OLE objects**
When generating PDF output, OLE objects are ignored, but space is reserved in the output for the objects.

### 3.3.4 Helpful tips

- The Reports Web Cartridge or CGI may create temporary files for report output that you may need to delete periodically. They will be located in the path specified by REPORT25_WEBLOC_TRANSLATED.

- For more information, see the demo Web reports, or refer to the Reports online help.

### 3.4 Guidelines for designing Graphics web applications
The following are tips for the efficient deployment of Graphics displays on the Web:

- Keep display file sizes small:
  - Limit the number of layers in a display.
  - Create objects programmatically.
  - Take advantage of stored procedures for data-intensive displays.

- Put as much code as possible into libraries to maximize code sharing between objects and applications and minimize file size during loading.

- Limit the use of multimedia within an application. Where you do use multimedia, redefine button triggers to make a call to a URL that contains media information.

- Limit activities that involve screen refreshing. For example, avoid the use of user interface-based timers. Design user interfaces so that events are initiated by user interaction rather than by timers.

- Take advantage of custom hyperlinks to create hyperlink drill-downs. For more information about custom hyperlinks, see Section 3.4.1, “Using custom hyperlinks in Graphics displays”.

---

*Guidelines for designing Graphics web applications*
Avoid the use of the following unsupported features when designing Graphics displays for the Web:

- Drag-and-drop.
- Mouse_Up trigger events. Only Mouse_Down is supported for this release.
- Multi-window applications. Instead, use layers rather than multiple windows, or use custom hyperlinks to link multiple displays in separate windows. For more information about using custom hyperlinks in Graphics displays, see Section 3.4.1, “Using custom hyperlinks in Graphics displays”.

### 3.4.1 Using custom hyperlinks in Graphics displays

The Developer/2000 Server provides the capability to enhance your Graphics displays through the use of custom hyperlinks. Custom hyperlinks might be used, for example, to design clickable links to mail messages. Or they might be used to design multi-frame HTML pages with clickable Tables of Contents that update the contents of adjoining frames. See the sample files displays provided with the Graphics Server for examples of how you can use custom hyperlinks in Graphics displays.

To include custom hyperlinks in a Graphics display, first create the object that will function as the hyperlink (for example, an underlined text item or a rectangle). In the object’s button procedure code, set the following parameters:

```plaintext
og_set_param ('OG_URL', 'http://my_url.com');
```

and, optionally:

```plaintext
og_set_param ('OG_URL_TARGET', my_target);
```

where `my_target` is one of the following text arguments:

- *(empty string)*: The URL specified replaces the contents of the current browser window (default behavior).
- `_self`: The URL specified displays within the current browser window.
- `frames [n]`: The URL specified displays within the nth frame in the current page.
- `_new`: The URL specified displays within a new browser window.

**Note:** The `OG_URL_TARGET` parameter is optional. If it is omitted, the URL specified in `OG_URL` is displayed within the current browser window.
Guidelines for designing Graphics web applications
This appendix covers these topics:

- Section A.1, “About Template HTML Files”
- Section A.2, “Working with the Cartridge HTML File Template”
- Section A.3, “Working with the Static (Non-Cartridge) HTML File Template”

A.1 About Template HTML Files

The Developer/2000 Server includes two “template” HTML files for your use in deploying Form Builder applications on the Web:

- CARTRIDGE.HTML (cartridge HTML file template)
- STATIC.HTML (static, non-cartridge HTML file template)

Any time you wish to base an HTML file on a template, simply copy the appropriate template to a new file name, then customize the new file to fit your application(s).

Note: Due to an HTML syntax requirement, you must locate the tags that define the Forms Client applet between the <BODY> and </BODY> tags of an HTML file.
A.2 Working with the Cartridge HTML File Template

When you deploy a Form Builder application on the Web in a cartridge implementation, you can create a cartridge HTML file for the application. The easiest way to do this is to modify the cartridge HTML file template:

```
ORACLE_HOME\forms45\demos\webdemos\cartridg.html
```

At runtime, the Forms Cartridge Handler dynamically creates a new HTML file by merging information from the following sources:

- The application’s cartridge HTML file
- The application’s cartridge settings
- The application’s URL

The new (dynamically-created) HTML file then downloads to the end user’s Web browser.

For more information, refer to the following sections:

- Section 2.1.3.3, “Creating an application cartridge”
- Section 2.1.3.6, “Step 3c: Providing URLs of your Web-enabled applications”
A.2.1 Snapshot: the cartridge HTML file template

Note: Carriage returns have been added to this example to make it more readable. When working with template files, do not add carriage returns. Refer to the online version of this file for more information.

```html
<html>
<!-- FILE: cartridg.html -->
<!-- Oracle Cartridge HTML File Template -->
<!-- Rename, and modify tags and parameter values as needed -->

<head><title>The Developer/2000 Server</title></head>

<body>
<br>Please wait while the Forms Client class files download and run.
<br>This will take a second or two...
<p>
<!-- applet definition (start) -->
<applet codebase="%LEASTLOADEDHOST%/codebase_vdir/"
    code="oracle.forms.uiClient.v1_4.engine.Main"
    archive="%LEASTLOADEDHOST%/jars_vdir/f45web.jar"
    height=20
    width=20>
    <param name="serverPort" value="9000">
    <param name="serverArgs"
        value="forms_param=%forms_param%
        user_param=%user_param%">
    <param name="serverApp" value="default">
</applet>
<!-- applet definition (end) -->
</body>
</html>
```
A.2.2 Modifying the cartridge HTML file template

Modify the Oracle cartridge HTML file template to fit your application as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>codebase</td>
<td>required Replace /codebase_vdir/ with the virtual directory you defined to point to the physical directory ORACLE_HOME\forms45\java. For example: /web_code/</td>
</tr>
<tr>
<td>archive</td>
<td>optional Replace or add to /jars_vdir/f45web.jar to provide the virtual directory path and file name of any JAR file(s) you want to be downloaded to end users’ Web browsers at application startup. For more information, refer to Section 2.1.4, “Configuring the Forms Client”.</td>
</tr>
<tr>
<td>serverPort</td>
<td>optional Replace 9000 with the number of the port on which the Forms Server Listener process was started. In most cases, the port number will remain 9000 (the default). For more information, refer to Section 2.1.2, “Starting and stopping the Forms Listener Server”.</td>
</tr>
<tr>
<td>serverArgs</td>
<td>optional Replace forms_param with any valid Form Builder command-line parameter. Replace user_param with any valid user-defined parameter.</td>
</tr>
<tr>
<td>Notes:</td>
<td></td>
</tr>
<tr>
<td>■ You can provide multiple Form Builder command-line and user-defined parameters.</td>
<td></td>
</tr>
<tr>
<td>■ You must provide a physical directory path for the .FMX file by including a directory path in the HTML file or by defining the FORMS45_PATH environment variable. The .FMX suffix is optional.</td>
<td></td>
</tr>
<tr>
<td>serverApp</td>
<td>optional Replace default with the name of your application class (if any). Use application classes for creating application-specific font mapping and icon path settings. For more information, refer to Section B.2, “Referencing an application class”.</td>
</tr>
<tr>
<td>leastloadedhost</td>
<td>optional During load balancing, this fixed-name place holder is replaced dynamically by the cartridge with the name of the least-loaded system. The cartridge gets this information from the Metrics Server at runtime.</td>
</tr>
</tbody>
</table>

Table A–1 Cartridge HTML file parameters and values
You can provide a delimited value for any cartridge HTML file parameter. For example, you could have the following line in your cartridge HTML file:

```
ARCHIVE="%Archive%"
```

You then must assign a value to `%Archive%` (either in the application’s cartridge settings, or in the application’s URL).

**Note:** All variable parameters in your cartridge HTML file must receive values in this way at runtime.

If a parameter does not receive a value, the Forms Cartridge Handler cannot build an HTML file to pass back to the user’s Web browser, resulting in an error. For information on checking for Forms Cartridge Handler errors, refer to Section 2.2, “Troubleshooting your Forms configuration”.

**Note:** If you are using load balancing and you use the ARCHIVE parameter to point to a physical directory where Oracle JAR files (and any custom JAR files) are stored, this URL must be the same as the URL for CODEBASE since everything must be downloaded from the same system. Thus, if you use `%LEASTLOADEDHOST%` in your CODEBASE definition, use it in your ARCHIVE definition as well. For example, if your old HTML file had the following entry for ARCHIVE:

```
ARCHIVE="jars_vdir/f45web.jar"
```

Then your new HTML file will have the following entry for ARCHIVE:

```
ARCHIVE="%LEASTLOADEDHOST%/jars_vdir/f45web.jar"
```
A.2.3 Examples of customized cartridge HTML files

The following examples display variations of the cartridge HTML file template.

A.2.3.1 Example 1
ACME has created a cartridge HTML file template. It includes the following:

- A title for the Web browser window
- A virtual directory for the Form Builder Java codebase: /web_code/
- A Form Builder command-line parameter: Module

Note: Carriage returns have been added to this example to make it more readable. When working with template files, do not add carriage returns. Refer to the online version of this file for more information.

```html
<HTML>
<!-- FILE: tempcart.html -->
<!-- ACME cartridge HTML file template -->
<!-- rename and modify as needed -->

<HEAD><TITLE>ACME Web Applications</TITLE></HEAD>

<BODY><CENTER>
<br>Welcome to ACME Web Applications!
<br>Please wait for the application to load...
<br></CENTER><P>

<APPLET CODEBASE="/web_code/"
    CODE="oracle.forms.uiClient.v1_4.engine.Main"
    HEIGHT="20" WIDTH="20">
<PARAM NAME="serverPort" VALUE="9000">
<PARAM NAME="serverArgs" VALUE="Module=%Module%">
</APPLET>

</BODY>

</HTML>
```
A.2.3.2 Example 2
General Traders Inc.’s cartridge HTML file includes the following:

- A virtual directory for images: imag
- A background image and main image: gti_wall.jpg and gti_logo.jpg
- A virtual directory for JAR files: /web_jars/
- Two JAR files: f45web.jar and gti.jar
- A virtual directory for the Form Builder Java codebase: /web_code/
- Two Form Builder command-line parameters: Module and Userid
- One user-defined parameter: Dept_id
- An application class setting: gti_ac
- The parameter to specify that load balancing be used: %leastloadedhost%

Note: Carriage returns have been added to this example to make it more readable. When working with template files, do not add carriage returns. Refer to the online version of this file for more information.

```html
<!-- FILE: gti_cart.html -->
<!-- General Traders cartridge HTML file -->

<BODY BACKGROUND="/imag/gti_wall.jpg">
<CENTER><IMG SRC="/imag/gti_logo.gif"
HEIGHT=170
WIDTH=120
ALT="gti_logo.jpg">
</CENTER><P>

<APPLET CODEBASE="%leastloadedhost%/web_code/"
CODE="oracle.forms.uiClient.v1_4.engine.Main"
ARCHIVE="/leastloadedhost%/web_jars/f45web.jar , /web_jars/gti.jar"
HEIGHT=20 WIDTH=20>
<PARAM NAME="serverPort" VALUE="7777">
<PARAM NAME="serverArgs" VALUE="Module=%Module% Userid=%Userid%
Dept_id=%Dept_id%">
<PARAM NAME="serverApp" VALUE="gti_ac">
</APPLET>
</BODY>
</HTML>
```
A.3 Working with the Static (Non-Cartridge) HTML File Template

When you deploy a Form Builder application on the Web in a non-cartridge implementation, you must create a static HTML file for the application. The easiest way to do this is to modify the Oracle static HTML file template:

```
ORACLE_HOME\forms45\demos\webdemos\static.html
```

At runtime, the static HTML file downloads to the end user’s Web browser.

For more information, refer to the following sections:

- Section 2.1.3.5, “Creating a static (non-cartridge) HTML file”
- Section 2.1.3.6, “Step 3c: Providing URLs of your Web-enabled applications”

A.3.1 Snapshot: the static (non-cartridge) HTML file template

Note: Carriage returns have been added to this example to make it more readable. When working with template files, do not add carriage returns. Refer to the online version of this file for more information.

```
<html>
<!-- FILE: static.html -->
<!-- Oracle Static (Non-Cartridge) HTML File Template -->
<!-- Rename, and modify tags and parameter values as needed -->

<head><title>Oracle</title></head>

<body><br>Please wait while the Forms Client class files download and run.
<br>This will take a second or two...
<br>
<!-- applet definition (start) -->
<applet codebase="/codebase_vdir/
    code=oracle.forms.uiClient.v1_4.engine.Main
    archive="/jars_vdir/f45web.jar"
    height=20
    width=20>
    <param name="serverPort" value="9000">
    <param name="serverArgs"
        value="module=fmx_name userid=user/password@datasource">
    <param name="serverApp" value="default">
</applet>

<!-- applet definition (end) -->
</body>
```

A-8 Developer/2000: Deploying Applications on the Web
A.3.2 Customizing the static (non-cartridge) HTML file template

Modify the Oracle static HTML file template to fit your application as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>codebase</td>
<td>required</td>
</tr>
<tr>
<td>archive</td>
<td>optional</td>
</tr>
<tr>
<td>serverPort</td>
<td>optional</td>
</tr>
<tr>
<td>serverArgs</td>
<td>optional</td>
</tr>
<tr>
<td>serverApp</td>
<td>optional</td>
</tr>
</tbody>
</table>

**Notes:**
- You can provide multiple Form Builder command-line and user-defined parameters.
- You must provide a **physical** directory path for the .FMX file by including a directory path in the HTML file or by defining the FORMS45_PATH environment variable. The .FMX suffix is optional.

Replace `forms_param` with any valid Form Builder command-line parameter. Replace `user_param` with any valid user-defined parameter.

Replace default with the name of your application class (if any). Use application classes for creating application-specific font mapping and icon path settings. For more information, refer to Section B.2, “Referencing an application class”.

*Table A–2  Static (non-cartridge) HTML file parameters and values*
A.3.3 Examples of customized static (non-cartridge) HTML files

The following examples provide variations of the static HTML file template.

A.3.3.1 Example 1
ACME’s static HTML file (for their ORDERS application) includes the following:

- A title for the Web browser window
- A brief welcoming message to users
- A virtual directory for the Form Builder Java codebase: /web_code/
- A virtual directory for JAR files: /web_jars/
- A JAR file: f45all.jar
- Two Form Builder command-line parameters: module and userid

Note: Carriage returns have been added to this example to make it more readable. When working with template files, do not add carriage returns. Refer to the online version of this file for more information.

```html
<html>
<head>
<title>ACME: Web Applications</title>
</head>
<body>
<center>
Welcome to ACME Web Applications!<br>
</center>
<br>
<!-- applet definition (start) -->
<applet
  codebase="/web_code/
  code="oracle.forms.uiClient.v1_4.engine.Main"
  archive="/web_jars/f45all.jar"
  height=20 width=20>
<br>
  <param name="serverPort" value="9000">
  <param name="serverArgs" value="module=c:\orant\bin\orders" userid=clerk@inventory">
</applet>
<!-- applet definition (end) -->
</body>
</html>
```

A.3.3.2 Example 2
General Traders Inc. has their own static HTML file template that includes the following:
Working with the Static (Non-Cartridge) HTML File Template

- A virtual directory for image: imag
- A background image and main image: gti_wall.jpg and gti_logo.jpg
- Form Builder command-line parameters that GTI changes for each application: module and userid

**Note:** Carriage returns have been added to this example to make it more readable. When working with template files, do not add carriage returns. Refer to the online version of this file for more information.

```html
<html>
<!-- FILE: gti_stat.html -->
<!-- GTI static HTML template -->
<!-- Rename and modify as needed -->
<body background="/imag/gti_wall.jpg">
<center><img src="/imag/gti_logo.jpg" height=170 width=120 alt="gti_logo.jpg"></center>
<p>
<applet codebase="/web_code/"
    code="oracle.forms.uiClient.v1_4.engine.Main"
    archive="/web_jars/gti.jar"
    height=20 width=20>
    <param name="serverPort" value="7777">
    <param name="serverArgs" value="module=c:\orant\bin\fmx_name userid=clerk@database">
</applet>
</p>
</body>
</html>
```
Including Application-Specific Settings

This appendix covers these topics:

■ Section B.1, “About Application Classes”
■ Section B.2, “Referencing an application class”
■ Section B.3, “Setting an icon directory path for an application class”
■ Section B.4, “Creating font-mapping settings for an application class”

B.1 About Application Classes

Application classes enable you to specify the following custom settings for your Web-enabled Forms applications:

■ The directory location of images files (in GIF format) for iconic buttons
■ Font mappings (application fonts to Java fonts)

To create settings for an application class, you must add the appropriate settings to the REGISTRY.DAT file:

ORACLE_HOME\forms45\java\oracle\forms\uiClient\v1_4\util\Registry.dat

Using application classes is optional, since Form Builder supplies default settings for font mappings and icon path. Use application classes if you wish to create a number of different settings that you can alternate between by switching a single reference in your application’s cartridge settings, cartridge HTML file, or static HTML file.

At runtime, Form Builder first looks for settings that correspond to the specified application class (if any). If no application class settings exist, Form Builder uses default settings.
B.2 Referencing an application class

To reference an application class for your application:

1. Include the `serverApp` parameter (and value) in your application’s cartridge settings, cartridge HTML file, or static HTML file.
   
   For example:
   ```xml
   <applet codebase="/web_code/"
       code="oracle.forms.uiClient.v1_4.engine.Runform"
       height="20" width="20">
       <param name="serverPort" value="9000">
       <param name="serverArgs" value="Module=inventory">
       <param name="serverApp" value="inv_form">
   </applet>
   ```

   **Note:** Carriage returns have been added to this example to make it more readable. When working with template files, do not add carriage returns.

2. Add font mapping and icon path settings (in the `REGISTRY.DAT` file) specific to the application.
   
   For more information refer to:
   
   - Section B.3, “Setting an icon directory path for an application class”
   - Section B.4, “Creating font-mapping settings for an application class”
B.3 Setting an icon directory path for an application class

To set an icon directory path for an application class:

1. Open the REGISTRY.DAT file in a text editor.
2. Add the following line to REGISTRY.DAT:
   
   <app_class>.iconpath=<path>

   Replace <app_class> with a valid application class referenced by your application, and replace <path> with one of the following:

   - A fully-qualified URL that points to any virtual directory on any server:
     
     inv_form.iconpath=http://abc_dev.com/html/icons/

   - A virtual directory on the same server from which the HTML page is downloaded:
     
     inv_form.iconpath=/icons/

   - A directory relative to the directory from which the HTML page was downloaded (non-cartridge implementations only):
     
     inv_form.iconpath=icons/

   **Note:** If you do not specify an icon path for an application class, Form Builder uses the default icon path setting in REGISTRY.DAT:

   default.iconpath=

   By default, Form Builder will search for iconic button image files in the directory from which the HTML file was downloaded.

   To specify the directory Oracle will search by default, append it to the Form Builder icon path setting. The path guidelines provided above still apply. For example:

   default.iconpath=http://abc_dev.com/html/icons/
B.4 Creating font-mapping settings for an application class

To create font-mapping settings for an application class:

1. Open the REGISTRY.DAT file in a text editor.
2. Add lines to the file as follows:

   `<app_class>.<font_param>=<param_value>`

Replace `<app_class>` with a valid application class, replace `<font_param>` with the name of the font parameter, and replace `<param_value>` with an appropriate value.

To map fonts from your application to their Java equivalents, add lines similar to:

   `inv_form.appFontnames=Courier,Terminal,Arial
   inv_form.javaFontnames=MonoSpaced,Dialog,SansSerif`

To map unrecognized application fonts to a Java font, add lines similar to:

   `inv_form.defaultFontname=Dialog
   inv_form.defaultSize=10
   inv_form.defaultStyle=BOLD`

Notes:

- The mapping of Java fonts to platform-specific fonts is automatic, according to default Java settings; you cannot alter the mappings. For more information, refer to Section 3.2.4, “Selecting fonts”.
- For more information on setting font preferences, refer to the commented sections of the REGISTRY.DAT file.
This appendix includes instructions for tasks involving Oracle WebServer 2.1. If you are using WebServer 2.1 instead of Web Application Server 3.0, use these instructions in place of the instructions in previous chapters. It includes:

- Section C.1, “Creating an Application Cartridge for Your Forms Server”
- Section C.2, “Setting Up the Reports Web Cartridge”
- Section C.3, “Creating the Graphics Server Cartridge Entry in the Web Request Broker”

C.1 Creating an Application Cartridge for Your Forms Server

Create an application cartridge for a Forms application by doing the following:

- Creating a cartridge
- Registering the cartridge with the Forms Cartridge Handler
- Creating a virtual directory for the cartridge
- Creating a base HTML file for the cartridge

C.1.1 Creating and registering the cartridge, and creating its virtual directory

The cartridge parameters and values that you set specify runtime settings of your Web application.

1. Install Oracle WebServer 2.1, if you haven’t already.

   Note: When you install the WebServer 2.1, use websvr as the name of the default Web Listener (on Windows NT, this is the default). You specify the listener name in the Oracle Web Listener Configuration dialog.
2. Start the Admin listener. For example, type:
   `wlct121 start admin`

3. Start the `websvr` listener. For example, type:
   `wlct121 start websvr`

4. Start your browser.

5. Access the Oracle Web Application Server by navigating to the appropriate URL (`http://<name of your machine or IP address>:<port number of Web Server Admin listener>`).
   For example: `http://myserver.com:9999`.
   The Oracle WebServer appears.

6. Click **WebServer Manager** to display the WebServer Administration page.

7. Click **Web Request Broker**.

8. Under the Modify Web Request Broker Configuration heading, select the appropriate Web server listener and click Modify.

9. To create a cartridge, under the Cartridges heading, enter a name for the new cartridge.
   For example: `web_cart`

10. Click **Create Cartridge**.
    The WRB Cartridge Configuration page appears.
11. Under the Cartridge Parameters heading, enter the following parameters and parameter values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseHTML(^1)</td>
<td>Enter the physical directory path and file name for the base cartridge HTML file accessed by the Forms Cartridge Handler at runtime. For information on creating a cartridge HTML file, refer to Section C.1.2, “Creating a base HTML file for the cartridge”.</td>
</tr>
<tr>
<td>HTMLdelimiter(^2)</td>
<td>Enter the one-character delimiter used to denote parameter values in the application’s cartridge HTML file. The delimiter can be any character. The default is % (percent sign).</td>
</tr>
<tr>
<td>Code</td>
<td>Enter the following: oracle.forms.uiClient.v1_4.engine.Main</td>
</tr>
<tr>
<td>Codebase</td>
<td>Enter the name of the virtual directory you defined to point to the physical directory ORACLE_HOME\forms45\java.</td>
</tr>
<tr>
<td>Archive</td>
<td>Provide the directory path and file name of any JAR file(s) you want downloaded to end users’ Web browsers at application startup. For more information, refer to Section 2.1.4.4, “Referencing JAR files”.</td>
</tr>
<tr>
<td>serverPort</td>
<td>Enter the number of the port on which the Forms Server Listener process was started. In most cases, the port is 9000 (the default). For more information, refer to Section 2.1.2.1, “Starting a Forms Server Listener process”.</td>
</tr>
<tr>
<td>&lt;forms_param&gt;</td>
<td>Enter valid Forms command-line parameters (such as module) and corresponding values.</td>
</tr>
<tr>
<td>&lt;user_param&gt;</td>
<td>Enter valid user-defined parameters and corresponding values.</td>
</tr>
<tr>
<td>serverApp</td>
<td>Enter the name of the application class you wish to apply to the application. Use application classes for creating application-specific font mapping and icon path settings. For more information, refer to the Section B.1, “About Application Classes”.</td>
</tr>
</tbody>
</table>

\(^1\) Parameters marked with asterisks (*) are required if using a cartridge HTML file.

Table C–1  Application cartridge parameters and values
Figure C–1  Setting application cartridge parameters and values
12. Click Modify Cartridge Configuration.
13. Click on the Go back to Web Request Broker Configuration for this listener link. Do not use the Back button on your Web browser to return to the Web Request Broker Configuration page, or the cartridge parameter will not be set.

14. To register your cartridge, under the Applications and Objects section, enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>The cartridge name. For example: web_cart</td>
</tr>
<tr>
<td>Object Path</td>
<td>The directory path and filename of the Forms Cartridge Handler.</td>
</tr>
<tr>
<td></td>
<td>▪ On Microsoft Windows NT: %ORACLE_HOME%\bin\f45webc.dll</td>
</tr>
<tr>
<td></td>
<td>▪ On UNIX: $ORACLE_HOME/lib/f45webc.so</td>
</tr>
<tr>
<td>Entry Point</td>
<td>form_entry</td>
</tr>
</tbody>
</table>

**Figure C–2  Forms Cartridge Handler process flow**

Application URL


Cartridge HTML File

```xml
<applet codebase="^Codebase^" code="^Code^"
archive="/jars/acme.jar"
width="20" height="20">
<param name="serverArgs"
value="Module=^Module^Deptid=^Deptid^">
... ...
```

Cartridge Settings

<table>
<thead>
<tr>
<th>setting</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseHTML</td>
<td>web_cart.html</td>
</tr>
<tr>
<td>HTMLdelimiter</td>
<td>^</td>
</tr>
<tr>
<td>Code</td>
<td>oracle.forms.uiCl</td>
</tr>
<tr>
<td>Codebase</td>
<td>/web_code/</td>
</tr>
<tr>
<td>serverPort</td>
<td>9000</td>
</tr>
</tbody>
</table>

Dynamically-Created HTML File (passed back to Web browser)
Creating an Application Cartridge for Your Forms Server

Figure C–3 Registering a cartridge with the Forms Cartridge Handler

15. Click Modify WRB Configuration.

16. To create a virtual directory for your cartridge, under the Applications and Directories section, enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min / Max</td>
<td>The minimum and maximum number of users, respectively, that can simultaneously connect to the cartridge.</td>
</tr>
<tr>
<td>Thread (T/P)</td>
<td>P Do not modify this field. This is an OWS21-specific field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Applications and Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>web_ca</td>
</tr>
</tbody>
</table>

**Figure C–3 Registering a cartridge with the Forms Cartridge Handler**

**15. Click Modify WRB Configuration.**

**16. To create a virtual directory for your cartridge, under the Applications and Directories section, enter values for the following fields:**

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Path</td>
<td>A virtual directory which—when included in an application’s URL—will automatically start its associated cartridge. Do not include a trailing slash. For example: /web_cart</td>
</tr>
<tr>
<td>Application</td>
<td>The name of the cartridge the WRB will execute whenever a URL containing the associated virtual directory is received by the WRB. For example: web_cart</td>
</tr>
<tr>
<td>Physical Path</td>
<td>The directory path for the Forms Cartridge Handler:</td>
</tr>
<tr>
<td></td>
<td>• On Microsoft Windows NT: %ORACLE_HOME%\bin\</td>
</tr>
<tr>
<td></td>
<td>• On UNIX: $ORACLE_HOME/lib/</td>
</tr>
</tbody>
</table>
Creating an Application Cartridge for Your Forms Server

Figure C–4  Creating a virtual directory for a cartridge

17. Click Modify WRB Configuration.

18. To activate all your changes stop and restart the websvr listener. Click the Listener Home button at the bottom of the WRB Administration page.

19. On the Oracle Web Listener Home Page, find the websvr listener and do the following:
   - Click Stop to stop the websvr listener.
   - Click Start to restart the websvr listener.

C.1.2 Creating a base HTML file for the cartridge

To create a base HTML file that the Forms Cartridge Handler will merge with information from your application’s cartridge settings and URL, create an application cartridge HTML file.

This is an optional step, and can be used for the following purposes:

- to include text or images on the HTML page that users see when they first start up a Web application
- to specify Forms command-line parameters (such as Module and Userid)
- to specify user-defined parameters

The easiest way to create an application cartridge HTML file is to modify the template provided by Oracle.

To create a base HTML file for the cartridge:

1. Make a copy of CARTRIDG.HTML (the Oracle template).
2. Rename the copy.
For example: cart.html

3. Open the file in a text editor and modify the text and applet tags as needed.

For information on the Oracle cartridge HTML file template (including a snapshot, instructions on modifying the template, and examples of customized files), refer to Section A.2, “Working with the Cartridge HTML File Template”.

4. Place the new file in the physical directory that corresponds to the virtual directory you defined for HTML files.

   Note: For information on creating a virtual directory to point to the physical directory where your HTML files are located, refer to Section 2.1.3.1, “Step 3a: Creating virtual directories on your Web server”.

C.2 Setting Up the Reports Web Cartridge

C.2.1 Setting Up for Windows 95/NT

To install the Web Cartridge on Windows 95/NT:

1. Install Developer/2000 Reports 2.5.

2. Install Oracle WebServer 2.1, if you haven’t already.

   Note: When you install the WebServer 2.1, use websvr as the name of the default Web Listener (on Windows NT, this is the default). You specify the listener name in the Oracle Web Listener Configuration dialog.

3. Start the Admin listener. For example, type:

   wlctl21 start admin

4. Start the websvr listener. For example, type:

   wlctl21 start websvr

5. Start your browser.

6. Access the Oracle Web Application Server by navigating to the appropriate URL (http://<name of your machine or IP address>:<port number of Web Server Admin listener>).

   For example: http://myserver.com:8888.

   The Oracle WebServer appears.

7. Click WebServer Manager to display the WebServer Administration page.

8. Click Web Request Broker.
9. Under the Modify Web Request Broker Configuration heading, select the appropriate Web server listener and click **Modify**.

10. Under the Cartridges heading enter a name for the new cartridge.
    For example: r25ows

11. Define the cartridge parameters REPORTS25_WEBLOC and REPORTS25_WEBLOC_TRANSLATED. For example, you could set REPORTS25_WEBLOC=/cache/ and REPORTS25_WEBLOC_TRANSLATED=c:\orant\mycache.

12. Click **Modify Cartridge Configuration**.

13. Click on the Go back to Web Request Broker Configuration for this listener link. Do not use the Back button on your Web browser to return to the Web Request Broker Configuration page, or the cartridge parameter will not be set.

14. In the Applications and Objects section, enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>The cartridge name. For example: r25ows</td>
</tr>
<tr>
<td>Object Path</td>
<td>The path for the cartridge. For example, if your $ORACLE_HOME is c:\orant, you would enter c:\orant\bin\r25ows32.dll.</td>
</tr>
<tr>
<td>Entry Point</td>
<td>r25ows_start</td>
</tr>
<tr>
<td>Min / Max</td>
<td>The minimum and maximum number of users, respectively, that can simultaneously connect to the cartridge.</td>
</tr>
</tbody>
</table>

15. Click **Modify WRB Configuration**.

16. In the Applications and Directories section, enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Path</td>
<td>A virtual directory which—when included in an application’s URL—will automatically start its associated cartridge. Do not include a trailing slash. For example: /r25ows</td>
</tr>
</tbody>
</table>
17. Click **Modify WRB Configuration**.

18. (Optional) Set up optional cartridge parameters and environment variables to obtain further functionality from R25OWS. Refer to the section on cartridge parameters and environment variables below.

19. To activate all your changes stop and restart the `websvr` listener. Click the **Listener Home** button at the bottom of the WRB Administration page.

20. On the Oracle WebListener Home Page, find the `websvr` listener and do the following:
   - Click **Stop** to stop the `websvr` listener.
   - Click **Start** to restart the `websvr` listener.

### C.2.2 Setting Up for UNIX

To install the Web Cartridge on UNIX:

1. Install Developer/2000 Reports 2.5.

2. Install Oracle Web Server 2.1, if you haven’t already.

   **Note:** When you install the Web Server 2.1, use `www` as the name of the default Web Listener (on UNIX, this is the default). You specify the listener name in the Oracle Web Listener Configuration dialog.

3. Start the Admin listener. For example, type:
   ```
   wlctl21 start admin
   ```

4. Start the `www` listener. For example, type:
   ```
   wlctl21 start www
   ```

5. Start your browser.
6. Access the Oracle Web Application Server by navigating to the appropriate URL (http://<name of your machine or IP address>:<port number of Web Server Admin listener>).
   For example: http://myserver.com:9999.
   The Oracle WebServer appears.

7. Click WebServer Manager to display the WebServer Administration page.

8. Click Web Request Broker.

9. Under the Modify Web Request Broker Configuration heading, select the appropriate Web server listener and click Modify.

10. Under the Cartridges heading enter a name for the new cartridge.
    For example: r25ows

11. Define the cartridge parameters REPORTS25_WEBLOC and REPORTS25_WEBLOC_TRANSLATED. For example, you could set REPORTS25_WEBLOC=/cache/ and REPORTS25_WEBLOC_TRANSLATED=/xyz/mycache.

12. Click Modify Cartridge Configuration.

13. Click on the Go back to Web Request Broker Configuration for this listener link. Do not use the Back button on your Web browser to return to the Web Request Broker Configuration page, or the cartridge parameter will not be set.

14. In the Applications and Objects section, enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>The cartridge name. For example: r25ows</td>
</tr>
<tr>
<td>Object Path</td>
<td>The path for the cartridge. For example, if your $ORACLE_HOME is /xyz/myhome, you would enter /xyz/myhome/r25owsm.so as the object path.</td>
</tr>
<tr>
<td>Entry Point</td>
<td>r25ows_start</td>
</tr>
<tr>
<td>Min / Max</td>
<td>The minimum and maximum number of users, respectively, that can simultaneously connect to the cartridge.</td>
</tr>
</tbody>
</table>

15. Click Modify WRB Configuration.
16. In the Applications and Directories section, enter values for the following fields:

<table>
<thead>
<tr>
<th>In this field...</th>
<th>Enter this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Path</td>
<td>A virtual directory which—when included in an application’s URL—will automatically start its associated cartridge. Do not include a trailing slash. For example: /r25ows</td>
</tr>
<tr>
<td>Application</td>
<td>The name of the cartridge the WRB will execute whenever a URL containing the associated virtual directory is received by the WRB. For example: r25ows</td>
</tr>
<tr>
<td>Physical Path</td>
<td>The directory path for the Reports Web Cartridge. For example, on Microsoft Windows NT: /xyz/myhome</td>
</tr>
</tbody>
</table>

17. Click **Modify WRB Configuration**.

18. (Optional) Set up optional cartridge parameters and environment variables to obtain further functionality from R25OWS. Refer to the section on cartridge parameters and environment variables below.

19. To activate all your changes stop and restart the WWW listener. Click the **Listener Home** button at the bottom of the WRB Administration page.

20. On the Oracle Web Listener Home Page, find the WWW listener and do the following:
   - Click **Stop** to stop the WWW listener.
   - Click **Start** to restart the WWW listener.
C.3 Creating the Graphics Server Cartridge Entry in the Web Request Broker

To create application cartridges, you must install and configure the Oracle Web Request Broker on your application server. Oracle WRB provides a framework for cartridges, and manages client connections to them.

C.3.1 Setting cartridge parameters and values

The cartridge parameters and values that you set specify runtime settings of your Web application.

To set cartridge parameters and values for an application:

1. Install Oracle WebServer 2.1, if you haven’t already.

   Note: When you install the WebServer 2.1, use websvr as the name of the default Web Listener (on Windows NT, this is the default). You specify the listener name in the Oracle Web Listener Configuration dialog.

2. Start the Admin listener. For example, type:
   `wlctl21 start admin`

3. Start the websvr listener. For example, type:
   `wlctl21 start websvr`

4. (UNIX only) Set the GRAPHICS_WEB_DIR and OWS_IMG_DIR environment variables.
   - Set GRAPHICS_WEB_DIR to the full directory path that contains the .OGDs that you want to run on the Web.
   - Set OWS_IMG_DIR to the location of the /ows-img/ virtual directory. You define the /ows-img/directory in the Oracle WebServer Administration page under Directory Mappings.

   Note: On UNIX, you must start the listener from the shell where you set the GRAPHICS_WEB_DIR and OWS_IMG_DIR environment variables.

   Note: On Windows NT, the Oracle Installer will set GRAPHICS_WEB_DIR to ORACLE_HOME\GRAPHICS25\WEB_OGD, and set OWS_IMG_DIR to ORACLE_HOME\OWS\3.0\ING. You can verify or change these values using regedit.

5. Start your browser.
6. Access the Oracle Web Application Server by navigating to the appropriate URL (http://<name of your machine or IP address>:<port number of Web Server Admin listener>).
   For example: http://myserver.com:9999.
   The Oracle WebServer appears.

7. Click **WebServer Manager** to display the WebServer Administration page.

8. Click **Web Request Broker**.

9. Under the Modify Web Request Broker Configuration heading, select the appropriate Web server listener and click **Modify**.

10. Under the Cartridges heading, enter a name for the new cartridge.
    For example: OGWEB

11. Click **Create Cartridge**.

    The WRB Cartridge Configuration page appears.

12. Under the Cartridge Parameters heading, enter the following parameters and parameter values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GW_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>The number indicating the time interval (in minutes) for which the display will remain open. When the time interval specified here has elapsed, the display is automatically closed.</td>
</tr>
</tbody>
</table>
GW_IMAGES_USE_FILES | YES

The value of this parameter determines whether or not the Graphics Server creates temporary GIF image files on the server (in the web_tmp directory in the img directory in the web server path) when returning HTML pages that show Graphics displays. If the value is NO, then the HTML page instead references the Graphics Server (a second time) to retrieve the image data, which is “streamed” back to the client by the server as binary content without creating a temporary file on the server.

The default value of this parameter (if it is omitted entirely) is YES.

**Note:** Creating temporary GIF files can introduce an administrative task of periodically removing these files from the server once it is certain that they are no longer being used. However, to the best of its ability, the Graphics cartridge automatically removes any old image files so as not to consume storage resources unnecessarily. The fact that this method (of using temporary files) can result in some “unused” image files on the server is offset by a small performance benefit, since using “streamed” images is slightly slower, due to the fact that the second, image request must also be routed through the WRB to the Graphics Server instead of being just an ordinary request to the server for an image file (which is an optimized operation).
Creating the Graphics Server Cartridge Entry in the Web Request Broker

### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GW_LINKS_CLOSE_PREV</td>
<td>YES</td>
</tr>
</tbody>
</table>
|                         | The value of this parameter determines whether or not the Graphics Server closes the current OGD file when navigating to another OGD file via an embedded hyperlink (i.e., URL).
|                         | If the value of this parameter is YES, and an embedded URL is used to open another OGD on any Graphics Server (i.e., if it contains the string “openfile”), then the current OGD will be closed and the new one opened.
|                         | This conserves resources on the server, since the user no longer needs the first OGD to remain open. Note that, of course, other types of URLs (e.g., to another web-page or HTML object) will operate as normal, and the OGD containing these links will remain open after executing them, regardless of the value of the above parameter.
|                         | Note, however, that if the user returns to the first OGD, either via the "back" button on their browser or via another URL, they will see a “fresh” version of this document -- any changes they made before will have vanished.
|                         | The default value of this parameter is YES, since it is usually most desirable to conserve server resources. However, whatever the value of this parameter, it’s behavior can always be overridden for special cases by setting the bind-variable OG_URL_CLOSE_THIS_DOC to either YES or NO in the PL/SQL function being used in the OGD to invoke the embedded URL (i.e. via OG_URL and OG_URL_TARGET). Note that the value of OG_URL_CLOSE_THIS_DOC must be explicitly set each time (before a link is executed) if the user wishes to override the value of GW_LINKS_CLOSE_PREV, since it is only meant for differentiating special cases from the system-defined behavior. |
| GW_WRITE_TRACE_FILE     | YES              |
|                         | The value of this parameter determines whether or not the Graphics Server writes a "trace" file to the server’s file-system.
|                         | The trace file can be used by an administrator for debugging purposes.
|                         | The name of this file is gwtrace.txt.
|                         | Since this information should only be required in special situations, the default value of this parameter is NO. |

13. Click **Modify Cartridge Configuration**.
14. Click on the Go back to Web Request Broker Configuration for this listener link. Do not use the Back button on your Web browser to return to the Web Request Broker Configuration page, or the cartridge parameter will not be set.

15. Under the Applications and Objects heading, enter the following value for each field:

<table>
<thead>
<tr>
<th>Field name</th>
<th>What you specify</th>
</tr>
</thead>
<tbody>
<tr>
<td>App.</td>
<td>OGWEB</td>
</tr>
<tr>
<td></td>
<td>The symbolic name for the cartridge. This is the name you assigned when you created the Web Request Broker entry for the cartridge.</td>
</tr>
<tr>
<td>Object Path</td>
<td>C:\ORANT\BIN\GW25W32.DLL</td>
</tr>
<tr>
<td></td>
<td>This is the full path and file name of the unique entry point for the Graphics Server.</td>
</tr>
<tr>
<td>Entry Point</td>
<td>GWWRBMain</td>
</tr>
<tr>
<td></td>
<td>This is the name of the unique entry point for the Graphics Server.</td>
</tr>
<tr>
<td>Min</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>The minimum number of processes that must be started for the cartridge. The recommended value for this field is 0 (zero).</td>
</tr>
<tr>
<td>Max</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>The maximum number of processes that will be allocated for the cartridge. When the number specified here is exceeded, the Web Request Broker spawns a new cartridge. The value you enter in this field will depend on the characteristics of your server machine. It is recommended that you set this value to 100 initially, although you may experiment with the value to tune performance.</td>
</tr>
<tr>
<td>Thread (T/P)</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Do not modify this field. This field is an OWS21-specific field.</td>
</tr>
</tbody>
</table>

16. Click Modify Cartridge Configuration.
17. Under the **Application and Directories** heading, enter the following value for each field:

<table>
<thead>
<tr>
<th>Field name</th>
<th>What you specify:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Path</td>
<td>A virtual directory which—when included in an application’s URL—will automatically start its associated cartridge. Do not include a trailing slash. For example: /ogweb</td>
</tr>
<tr>
<td>App.</td>
<td>OGWEB</td>
</tr>
<tr>
<td></td>
<td>The symbolic name for the cartridge. This is the name that you assigned when you created the Web entry for the cartridge.</td>
</tr>
<tr>
<td>Physical Path</td>
<td>The path portion of the “Object Path” you specified under the “Applications and Objects” heading. For example, ORACLE_HOME/BIN.</td>
</tr>
</tbody>
</table>

18. Click **Modify Cartridge Configuration**.

19. To activate all your changes stop and restart the **websvr** listener. Click the **Listener Home** button at the bottom of the WRB Administration page.

20. On the Oracle Web Listener Home Page, find the **websvr** listener and do the following:

   - Click **Stop** to stop the **websvr** listener.
   - Click **Start** to restart the **websvr** listener.

21. (UNIX only) Before running the cartridge, create the directory $OWS_IMG_DIR/web_tmp, then give it global write permissions.

   For example, type:

   ```
   mkdir $OWS_IMG_DIR/web_tmp
   chmod 777 $OWS_IMG_DIR/web_tmp
   ```

   **Note:** On Windows NT, the Oracle Installer will create this directory automatically.
Applet
A small Java program that can be dynamically imported into Web pages or applications as needed.

CGI — Common Gateway Interface
The industry-standard technique for running applications on a Web server. Whereas standard HTML documents retrieved from a Web server are static (the exact same text is retrieved every time), CGI enables a program running on the Web server to communicate with another computer to generate “dynamic” HTML documents in response to user-entered information.

Designer/2000
Oracle’s application modelling, design and generation toolset. Using Designer/2000, you can generate forms and menus from module definitions recorded in the Designer/2000 Repository.

Encryption
The practice of scrambling (encrypting) data in such a way that only an intended recipient can unscramble (decrypt) and read the data.

Firewall
A computer that regulates access to computers on a local area network from outside, and regulates access to outside computers from within the local area network (LAN).
**Forms Cartridge Handler**
The component that dynamically creates HTML files (for Web-enabled Forms applications) at runtime by merging information from the application’s cartridge HTML file, cartridge parameter settings, and URL.

**Forms Client**
A Java applet—downloaded at runtime from an application server to an end user’s Web browser—that displays the form’s user interface and manages interaction between end users and the Forms Server.

**Forms Server Listener**
The Forms Server component that initiates the Forms runtime session and establishes a connection between the Forms Client and the Forms Server Runtime Engine.

**Forms Server Runtime Engine**
A modified version of the Forms 4.5 Runtime Engine, with user interface functionality redirected to the Forms Client. It handles all form functionality except UI interaction, including trigger and commit processing, record management, and general database interaction.

**GUI — Graphical User Interface**
The use of pictures rather than just words to represent the input and output of a program. Programs with GUIs run under a windowing system (such as X Windows, Microsoft Windows, Apple Macintosh, and so on). GUI programs display icons, buttons, and so on, in windows on the screen; users control the GUI programs mainly by moving a pointer on the screen (typically controlled by a mouse).

**HTML — Hypertext Markup Language**
A tag-based ASCII language used to specify the content and hypertext links to other documents on Web servers on the Internet. End users with Web browsers view HTML documents and follow links to display other documents.

**HTTP — Hypertext Transfer Protocol**
The protocol used to carry network traffic between a Web browser computer and the Web server being accessed.
**Hyperlink**
A reference (link) from some point in one hypertext document to (some point in) another document or another place in the same document. A Web browser usually displays a hyperlink in some distinguishing way (in a different color, font or style). When users activate hyperlinks (by clicking on them with a mouse) the browser displays the target of the link.

**Hypertext**
A collection of documents containing cross-references which, with the aid of a Web browser, allow readers to move easily from one document to another.

**Internet**
A worldwide TCP/IP-based network of computers.

**Intranet**
An internal TCP/IP network, access to which is restricted (via a firewall) to individuals inside the company or organization. An intranet provides similar services within an organization to those provided by the Internet, but is not necessarily connected to the Internet. A common example of an intranet is when a company sets up one or more Web servers on an internal network for distribution of information or applications within the company.

**IP (Internet Protocol) Address**
A four-part number with no more than three digits in each part, that uniquely identifies a computer on the Internet.

**JAR — Java Archive File**
A single compressed file comprised of many files (Java class files, images, and so on).

**Java**
A computer language that supports programming for the Internet in the form of platform-independent “applets.”

**NCA—Network Computing Architecture**
A common set of technologies that will allow all PCs, network computers and other client devices to work with all Web servers, database servers and application servers over any network.
ODBC—Open Database Connectivity

A standard for accessing different database systems. An application can submit statements to ODBC using the ODBC flavor of SQL. ODBC then translates these to whatever flavor the database understands. Using the Oracle Open Client Adapter (OCA), an application can access different database management systems in one consistent manner. This allows an application developer to develop, compile, and ship an application without targeting a specific DBMS.

ORACLE_HOME

An environment variable that indicates the root directory of Oracle products.

PDF — Portable Document Format

A file format (native for Adobe Acrobat) for representing documents in a manner that is independent of the original application software, hardware, and operating system used to create the documents. A PDF file can describe documents containing any combination of text, graphics, and images in a device-independent and resolution independent format.

PL/SQL

Oracle’s proprietary extension to the SQL language. Adds procedural and other constructs to SQL that make it suitable for writing applications.

Port

A number that TCP uses to route transmitted data to and from a particular program.

RC4

A cipher designed by RSA Data Security, Inc. which can accept keys of arbitrary length, and is essentially a pseudo random number generator with the output of the generator being XORed with the data stream to produce the encrypted data. For this reason, it is very important that the same RC4 key never be used to encrypt two different data streams.

RDBMS — Relational Database Management System

A database that allows the definition of data structures, storage and retrieval operations, and integrity constraints. In such a database, data and relations between them are organized in tables.
SNS/ANO—Secure Network Services Advanced Networking Option
An Oracle SQL*Net feature for providing data integrity, data privacy, authentication, and authorization for transmissions to and from the Oracle7 database. For more information, refer to the Oracle Advanced Networking Option Administrator’s Guide.

Socket
The combination of an IP address and a port number.

TCP — Transmission Control Protocol
The underlying communication protocol for exchanging HTTP requests between clients and Web servers.

URL: Uniform Resource Locator
The “address” used to specify a WWW server and home page. For example:

http://www.acme.com/

which indicates that the host’s address is www.acme.com.

A URL (often pronounced “Earl”) often is a filename (possibly with a long directory path), usually with an extension of .HTML, or .HTM (for PC-DOS filenames).

Virtual Directory
A directory mnemonic (essentially a shortcut or synonym) that the virtual file system uses to map files stored in the file system on the host machine.

Virtual File System
A mapping that associates the pathnames used in URL to the file system maintained by the host machine’s operating system.

Web
See WWW—World Wide Web.

Web Browser
A program that end users utilize to read HTML documents and programs stored on a computer (serviced by a Web server).

Web Cartridge
A program executed (on an application server) by the Web Request Broker.
Web Server (aka Web Listener)
A server process (HTTP daemon) running at a Web site which “listens” for HTTP requests and responds by displaying Web pages, downloading files, and so on, for the user who send the request.

WRB — Web Request Broker
Oracle’s powerful distributed runtime environment for developing and deploying applications for the Web. The WRB runtime platform enables application developers to write applications that are independent of, and work with a number of, Web servers.

WWW — World Wide Web
The network of servers on the Internet, each of which has one or more home pages, which provide information and hypertext links to other documents on that and (usually) other servers.
Index

A

action
  in Web report, 3-14
applet see Java applet
application
  Web design guidelines, 3-1
application cartridge HTML files
  about, C-7
  accessing See URL
  creating, C-7
  examples, A-6
  Oracle template, about, A-2
  Oracle template, modifying, A-4
  Oracle template, snapshot of, A-3
  setting parameter values in, A-4
  specifying as a cartridge parameter, 2-9, C-3
application cartridges
  glossary term, Glossary-5
  HTML files See application cartridge HTML files
  setting parameters and values, C-1, C-13
application class
  creating, B-1
  creating font-mapping settings, B-4
  setting icon directory path, B-3
application classes
  about, B-1

C

cartridge HTML file, A-2
cartridge parameters
  REPORTS25_OWSDIAGBODYSARGS, 2-40
  REPORTS25_OWSDIAGHEADARGS, 2-40
  REPORTS25_OWSHELP, 2-39
  REPORTS25_OWSMAP, 2-40
  REPORTS25_OWSNODIAG, 2-41
  REPORTS25_OWSPATHONLYURL, 2-40

CGI (Common Gateway Interface)
  configuring for Report Builder, 2-33
  dynamic reporting with, 1-11
  environment variables for Report Builder, 2-37
  glossary term, Glossary-1
  specifying URL report run requests, 2-42

Common Gateway Interface See CGI
  creating
    Graphics Server cartridge entry in the WRB, C-13
    HTML files (cartridge), C-7

D

Designer/2000
  glossary term, Glossary-1
  using with Developer/2000, 3-5
documentation
  related documents, xii
typographic conventions, xii
drill-down reporting, 1-4
dynamic reporting, 1-4

E

encryption
  glossary term, Glossary-1
Web applications, 1-9
environment variables
  LD_LIBRARY_PATH, 2-38
ORACLE_HOME, 2-38
REPORTS25_CGIDIAGBODYTAGS, 2-41
REPORTS25_CGIDIAGHEADTAGS, 2-42
REPORTS25_CGIHLP, 2-41
REPORTS25_CGIMAP, 2-41
REPORTS25_CGINODIAG, 2-42
REPORTS25_CGIPATHONLYURL, 2-42
REPORTS25_OWSDIAGBODYTAGS, 2-40
REPORTS25_OWSDIAGHEADTAGS, 2-40
REPORTS25_OWSHELP, 2-39
REPORTS25_OWSMAP, 2-40
REPORTS25_OWSNODIAG, 2-41
REPORTS25_OWSPATHONLYURL, 2-40
REPORTS25_WEBLOC, 2-38
REPORTS25_WEBLOC_TRANSLATED, 2-39
FORMS45_OUTPUT, 3-2
FORMS45_REPFORMAT, 3-2

G
Graphics
configuring the WRB, C-13
Graphics Builder
running from Form Builder, 3-3
Web applications
architecture, 1-14
configuring, 2-49
design guidelines, 3-20
Graphics Client, 1-15
Graphics Server, 1-15
GUI (Graphical User Interface)
glossary term, Glossary-2
GW_IMAGES_USE_FILES, 2-52
GW_LINKS_CLOSE_PREV, 2-53
GW_WRITE_TRACE_FILE, 2-53

H
HTML (Hypertext Markup Language)
application files (cartridge) See Application
cartridge HTML files
application files (non-cartridge) See non-
cartridge application HTML files
glossary term, Glossary-2
report output, 3-16
template files, A-1
template HTML files, about, A-1
HTML file, 2-22
HTTP (Hypertext Transfer Protocol)
glossary term, Glossary-2
redirection, report output and, 1-14
hyperlink
glossary term, Glossary-3
report output see Reports Web output
hypertext
glossary term, Glossary-3
see hyperlink

Index-2
Index-3

I
images
  in HTML report output,  3-18
Internet,  1-1
intranet,  1-1

J
JAR files,  2-13
glossary term,  Glossary-3
Java
  applet see Java applet
class files see Java class files
glossary term,  Glossary-3
Java applet,  1-5
glossary term,  Glossary-1
JAVA fonts,  3-6

L
load balancing
cartridge base HTML file,  2-22
deployment example,  2-16
Metrics Server and Client installation,  2-21
requirements,  2-18
setting up,  2-15
web cartridge configuration,  2-19
LRS
  See Log roll-forward server (LRS),  A-1

M
mapping
  application fonts to Java fonts,  B-4
Metrics Client
  installation,  2-21
  starting,  2-23
Metrics Server
  installation,  2-21
  starting,  2-22
modifying
  Oracle application cartridge HTML file template,  A-4
  Oracle non-cartridge application HTML file template,  A-9

N
NCA (Network Computing Architecture)
glossary term,  Glossary-3
Network Computing Architecture (NCA),  1-3
non-cartridge application HTML files
deployment examples,  A-10
Oracle template, about,  A-8
Oracle template, modifying,  A-9
Oracle template, snapshot of,  A-8
non-cartridge HTML file template
  See static HTML file template

O
ODBC (Open Database Connectivity)
glossary term,  Glossary-4
Oracle Web Request Broker,  1-15, 2-6

P
parameters
  GW_IMAGES_USE_FILE,  2-52
  GW_IMAGES_USE_FILES,  C-15
  GW_LINKS_CLOSE_PREV,  2-53, C-16
  GW_TIMEOUT,  2-52, C-14
  GW_WRITE_TRACE_FILE,  2-53, C-16
PDF (Portable Document Format),  3-19
glossary term,  Glossary-4
port
  glossary term,  Glossary-4
  specifying as a cartridge parameter,  2-9, C-3

R
RC4
glossary term,  Glossary-4
RDBMS (Relational Database Management System)
glossary term,  Glossary-4
RDF files
deployment,  2-46
REGISTRY.DAT file
  about,  B-1
  setting icon directory path in,  B-3
Relational Database Management System
  see RDBMS

Index-3
Report Builder
configuring for the Web, 2-25
escapes, 3-14
HTML output, about, 3-16
output format, 3-15
tips, 3-20
Web applications
architecture, 1-10
configuring, 2-25
design guidelines, 3-8
environment variables, 2-37
troubleshooting, 2-47
reports
actions, about, 3-14
adding Web functionality, 3-9
backgrounds, in HTML output, 3-17
bookmarks, about, 3-13
bookmarks, in HTML output, 3-17
buttons, in HTML output, 3-17
buttons, in PDF output, 3-19
drawings, in HTML output, 3-17
escapes, about, 3-14
fonts, in HTML output, 3-17
frames, in HTML output, 3-17
hyperlinks, about, 3-9
images, in HTML output, 3-18
OLE objects, in HTML output, 3-18
OLE objects, in PDF output, 3-20
overlapping objects, in HTML output, 3-18
pagination, in HTML output, 3-18
PDF output, about, 3-19
Reports Queue Manager, 1-4
Reports Server
about, 1-12
Reports Web Cartridge
about, 1-12
configuring, 2-33
dynamic reporting with, 1-11
environment variables, 2-37
specifying URL report run requests, 2-42
Reports Web CGI
about, 1-12
RSA RC4 40 bit encryption, 1-9
RUN_PRODUCT, 3-2

S
Secure Network Services Advanced Networking
Option see SNS/ANO
session IDs, 1-16
setting
application cartridge parameters and values, C-1, C-13
Graphics Server parameters in the WRB, C-13
parameter values for application cartridge
HTML files, A-4
settings
for Web-enabled Form Builder applications, B-2
SHOW_DOCUMENT, 3-6
SNS/ANO (Secure Network Services Advanced Networking Option)
glossary term, Glossary-5
socket
glossary term, Glossary-5
specifying
cartridge HTML file as a cartridge parameter, 2-9, C-3
port as a cartridge parameter, 2-9, C-3
SQL*Net SNS/ANO, 1-9
static HTML file template

T
template HTML files, A-1
templates
HTML file templates See application cartridge
HTML files and Non-cartridge application
HTML files
See application cartridge HTML files
See Non-cartridge application HTML files
thin client, 1-3
three-tiered architecture, 1-2

U
URL
parameterized, 1-17
specifying report run requests, 2-42
URL (Uniform Resource Locator)
example report run requests, 2-43
glossary term, Glossary-5
mapping for report run requests, 2-45
using
   WRB (Web Request Broker), C-13

V
virtual directories, 2-5
   glossary term, Glossary-5

W
web applications
   classes See application classes
web browser
   glossary term, Glossary-5
web cartridge see application cartridges
Web cartridges, 1-3
web listener see Web server
Web publishing, 1-4
Web Request Broker See WRB
Web server
   glossary term, Glossary-6
World Wide Web
   architecture
      Form Builder, 1-4
      Graphics Builder, 1-14
      Report Builder, 1-10
   configuring for Web applications, 2-1
   deploying applications
      Form Builder, 1-8
      overview, 1-1
   designing web applications, 3-1
   troubleshooting
      Form Builder, 2-25
WRB (Web Request Broker)
   about, C-13
   creating application cartridges with, 2-7, 2-33, 2-49, C-1, C-2, C-8, C-10, C-11, C-13, C-14
   glossary term, Glossary-6
   using, C-13