Oracle® Reports Developer

Publishing Reports

Release 6i

Part No. A73173-01

April 2000
## Contents

Send Us Your Comments ........................................................................................................................................ x

Preface ........................................................................................................................................................................ xiii

- Reports Server New Features ................................................................................................................................... xiii
- Intended Audience ....................................................................................................................................................... xiv
- Structure ......................................................................................................................................................................... xiv
- Related Documents ....................................................................................................................................................... xv
- Notational Conventions ................................................................................................................................................ xv

### Part I  Publishing Reports

#### 1  Introduction

1.1  The Oracle Reports Server ..................................................................................................................................... 1  
1.1.1  Oracle Reports Server ........................................................................................................................................ 2

#### 2  Publishing Architecture and Concepts

2.1  The Oracle Reports Server ..................................................................................................................................... 3  
2.2  Reports Server Architecture ................................................................................................................................... 4  
2.2.1  Web architecture: server configurations ........................................................................................................... 5  
2.2.1.1  Processing Web reports ................................................................................................................................ 6  
2.2.2  Non-Web architecture: server configuration ....................................................................................................... 7  
2.2.2.1  Processing reports ................................................................................................................................................ 8
## 2.3 Reports Server Configuration Choices
- Enable Web and non-Web requests
- Choose the Reports Web CGI or Servlet
- Choose the location of the Reports Server

## 3 Configuring the Reports Server
- Installation and Configuration Overview
- Configuring the Reports Server automatically
- Configuring the Reports Server manually
- Configuring the Reports Server: Reports Web CGI Example

### 3.1 Installation and Configuration Overview
- Configuring the Reports Server automatically

#### 3.1.1 Step 1: Starting the Oracle Installer

#### 3.1.2 Step 2: Choosing the Oracle Installation Settings

#### 3.1.3 Step 3: Choosing the Oracle Tools Installation Options

#### 3.1.4 Step 4: Selecting the Installation Options

#### 3.1.4.1 Typical install

#### 3.1.4.2 Custom install

### 3.2 Configuring the Reports Server: Reports Web CGI Example
- Step 1. Installing the software
- On the Web server machine
- On the Reports Server machine
- Step 2. Configuring the Web server
- Step 3. Configuring the Reports Web CGI
- Configuring Reports Web CGI
- Creating a service entry for the Reports Server
- (Optional) Setting the default Reports Server
  - For Windows NT
  - For UNIX
- Step 4. Configuring the Reports Server
- Configuring the Reports Server in Windows NT
- (Optional) Setting the source path
- Starting the Reports Server
- Configuring the Reports Server on UNIX
- (Optional) Setting environment variables
- Starting the Reports Server
4 Running Report Requests

4.1 Report Request Methods ................................................................. 33
4.2 Duplicate Job Detection ............................................................... 34
4.3 Using a Key Map File ................................................................. 35
4.3.1 Enabling key mapping ............................................................ 36
4.3.2 Mapping URL parameters .................................................... 37
4.4 Specifying Report Requests ....................................................... 37
4.4.1 Building a report ................................................................. 37
4.4.2 Specifying a report request from a Web browser ....................... 38
4.4.3 Scheduling reports to run automatically ................................. 39
5 Controlling User Access to Reports

5.1 Access Control Configuration and Setup Overview .......................................................... 42
5.2 Installing and Configuring Reports Security in WebDB ................................................... 43
  5.2.1 Step 1. Configure reports security in WebDB .............................................................. 43
     5.2.1.1 To install WebDB................................................................................................... 43
     5.2.1.2 To install the Reports Server security packages in WebDB ................................ 44
     5.2.1.3 To set the authentication cookie domain............................................................ 45
5.3 Setting Up Access Controls in WebDB ............................................................................. 45
  5.3.1 Step 1. Configure the Reports Server for access control ............................................ 47
     5.3.1.1 To create the TNS names alias that connects to WebDB .................................... 47
     5.3.1.2 To restrict access to the Reports Server............................................................ 48
  5.3.2 Step 2. Create user accounts....................................................................................... 49
     5.3.2.1 To create the Reports Developer system administrator user account............ 50
     5.3.2.2 To create users accounts for running reports..................................................... 50
  5.3.3 Step 3. Create availability calendars ........................................................................... 51
     5.3.3.1 To create the daily calendar................................................................................ 52
     5.3.3.2 To create the maintenance calendar ................................................................. 53
     5.3.3.3 To create the Christmas calendar...................................................................... 54
     5.3.3.4 To create a combined availability calendar....................................................... 55
  5.3.4 Step 4. Add access to a Reports Developer printer in WebDB .................................. 56
  5.3.5 Step 5. Add access to the Reports Server in WebDB ............................................... 57
  5.3.6 Step 6. Add access to the report definition file in WebDB ......................................... 58
     5.3.6.1 To create a list of values for the lastname user parameter............................... 59
     5.3.6.2 To add access the report definition file.............................................................. 61
  5.3.7 Step 7. Set parameter values on the Reports Developer parameter form ................ 63
     5.3.7.1 To run the report output to cache....................................................................... 64
     5.3.7.2 (optional) To run the report output to a restricted printer ................................. 65
     5.3.7.3 To set the default parameters for users at runtime .......................................... 66
  5.3.8 Step 8. Make the report available to users................................................................... 67
     5.3.8.1 To create a WebDB site....................................................................................... 67
     5.3.8.2 To create a folder in the WebDB site.................................................................. 67
     5.3.8.3 To add the report request to the folder............................................................... 69
     5.3.8.4 To run this report as a user................................................................................ 69
Part II  Appendices

A  RWCLI60 Command Line Arguments

B  Reports Server Configuration Parameters

C  Environment Variables

D  Database Connection Strings

E  Reports Web Cartridge

E.1  Reports Web Cartridge Example ................................................................. 215
E.1.1  Step 1. Install the software ............................................................... 215
E.1.2  Step 2. Configure your Web server ................................................. 217
E.1.3  Step 3. Configure the Reports Web Cartridge ....................................... 217
E.1.3.1  To create a service entry for the Reports Server ...................... 217
E.1.3.2  To set the default Reports Server .............................................. 218
E.1.3.3  To configure the Reports Web Cartridge (on Windows NT and UNIX) ...... 219
E.1.4  Step 4. Configure the Reports Server .................................................. 220
E.1.4.1  To configure the Reports Server on Windows NT ....................... 220
E.1.4.1.1  (Optional) Set the source path ............................................... 220
E.1.4.2  To configure the Reports Server on Windows NT ....................... 221
E.1.4.2.1  (Optional) Set the source path ............................................... 221
E.1.4.2.2  Install the Reports Server on Windows NT as a service .......... 221
E.1.4.2.3  Configure the Reports Server on Windows NT as a non-service ...... 222
E.1.4.2.4  Uninstall the Reports Server on Windows NT ....................... 223
E.1.4.3  To configure the Reports Server on UNIX .................................... 223
E.1.4.3.1  (Optional) Set environment variables ..................................... 223
E.1.4.3.2  Starting the Reports Server ..................................................... 224
Send Us Your Comments

Oracle Reports Developer Publishing Reports, Release 6i
Part No. A73173-01

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 (if available). You can send comments to us in the following ways:

- E-mail - oddoc@us.oracle.com
Preface

This manual describes the different options available for publishing reports with Oracle Reports Developer as well as how to configure the Reports Server software for publishing reports.

**Reports Server New Features**

<table>
<thead>
<tr>
<th>New Feature</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database queue.</strong> Update your database with Reports Server queue activity.</td>
<td>Section 3.2.6.1, “Updating the database with job queue activity”</td>
</tr>
<tr>
<td><strong>Reports Servlet.</strong> Configure the Reports Server using Java-enabled Web servers.</td>
<td>Section 3.4, “Reports Servlet Configuration Guidelines”</td>
</tr>
<tr>
<td><strong>Control user access to report.</strong> Restrict user access to reports that are run on Reports Servers. Oracle Reports uses Oracle WebDB to check that users have the necessary access privileges to run the report on restricted Reports Servers.</td>
<td>Chapter 5, “Controlling User Access to Reports”</td>
</tr>
<tr>
<td><strong>Clustering.</strong> Improve performance and loading balancing by clustering your Reports Servers.</td>
<td>Chapter 6, “Configuring Reports Server Clusters”</td>
</tr>
<tr>
<td><strong>Report Customization.</strong> Apply customizations to reports at runtime without changing the original report and generate different output depending upon the audience.</td>
<td>Chapter 7, “Customizing Reports at Runtime”</td>
</tr>
</tbody>
</table>
**Intended Audience**

This manual is intended for anyone who is interested in publishing reports with Oracle Reports Developer. It could be that you have built reports yourself and now want to publish them to a wider audience in your organization. It could also be that someone else built the reports for you and you now want to deploy them for other users to access. To configure Oracle Reports Developer software for publishing reports, you should have a thorough understanding of the operating system (for example, Windows NT or Solaris) as well as Net8. If you are planning to deploy reports dynamically on the Web, you should also be knowledgeable about your Web server configuration.

**Structure**

This manual contains the following chapters and appendices:

- **Chapter 1** Introduces you to the Reports Server.
- **Chapter 2** Introduces the architecture of the Reports Server and choices that you need to make before you configure the Report.
- **Chapter 3** Describes how to configure the Reports Server with example configurations using a cgi-aware Web server.
- **Chapter 4** Describes the various methods for running reports to the Reports Server.
- **Chapter 5** Describes how the Reports Server can be integrated with Oracle WebDB to control user access to reports.
- **Chapter 6** Describes how to configure the Reports Server with clustering to enhance performance and reliability.
- **Chapter 7** Describes how to use XML to apply customizations to reports at runtime.
Related Documents

For more information on building reports, Oracle WebDB, or the Oracle Report Server, refer to the following manuals:

- *Oracle Reports Developer Building Reports*, A73172-01
- *Oracle Reports Developer Getting Started for Windows*, A73156-01
- *Oracle WebDB Getting Started-Installation and Tutorial*, A70070-01
- *Deploying Forms to the Web with Oracle Developer Server*, A73071-01

Notational Conventions

The following conventions are used in this book:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface text</strong></td>
<td>Used for emphasis. Also used for menu items, button names, labels, and other user interface elements.</td>
</tr>
<tr>
<td>italicized text</td>
<td>Used to introduce new terms.</td>
</tr>
<tr>
<td>courier font</td>
<td>Used for path and file names, and for code and text that you type.</td>
</tr>
<tr>
<td>COURIER CAPS</td>
<td>Used for: File extensions (.PLL or .FMX) SQL commands</td>
</tr>
<tr>
<td>CAPS</td>
<td>Used for: Environment variables Built-ins and package names Executable names</td>
</tr>
</tbody>
</table>
Part I

Publishing Reports

Chapter 1, "Introduction"
Chapter 2, "Publishing Architecture and Concepts"
Chapter 3, "Configuring the Reports Server"
Chapter 4, "Running Report Requests"
Chapter 5, "Controlling User Access to Reports"
Chapter 6, "Configuring Reports Server Clusters"
Chapter 7, "Customizing Reports at Runtime"
In today’s fast-moving, competitive business world, clear and up-to-date information is needed for the accurate, expedient decision making requirements of an often geographically distributed workforce. The timely distribution of that information must be reliable, cost effective, and accessible to everyone who requires it. Reports Developer provides an unbounded, easy-to-use, scalable, and manageable solution for high-quality database publishing and reporting.

Reports Developer is a powerful enterprise reporting tool used by IS developers to create sophisticated dynamic reports for the Web and across the enterprise.

Reports Developer’s server-based architecture means report consumers require only a Web browser to view reports in industry standard formats. The Oracle Reports Server supports on-demand delivery of high-quality reports over the Web through native generation of HTML with Cascading Style Sheets and Adobe’s Portable Document Format (PDF). Maintenance overhead costs are cut as reports are administered and maintained centrally and there is no requirement to install complex software on every user’s PC.

1.1 The Oracle Reports Server

The Reports Server enables you to implement a multi-tiered architecture for running your reports. With the Reports Server, you can run reports on a remote application server.

When used in conjunction with the Reports Web CGI or Reports Servlet, the Reports Server also enables you to run reports from a Web browser using standard URL syntax. The Reports 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
The Oracle Reports Server

The Oracle Reports 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.

The Reports Server keeps track of a predefined maximum number of past jobs. Information on when the jobs are queued, started, and finished is kept, as well as the final status of the report. This information can be retrieved and reviewed on Windows from the Reports Queue Manager (RWRQM60) or via the API. The Reports Queue Manager may reside on the same machine as the Reports Server or on a client machine. On UNIX, you can use the Reports Queue Viewer (RWRQV60) to view the Reports Server’s queue.

1.1.1 Oracle Reports Server

The Reports Server is one of the components of Oracle Reports Server, a new generation of development tools that enable you to deploy new and existing Oracle Reports Developer reports on an internal company intranet, external company extranet, or on the Internet.

In addition to the Reports Server, the Oracle Reports Server includes the Graphics Server. The Graphics Server assists with the deployment of Oracle Graphics. Graphics are deployed as Graphic Interchange Format (.GIF) files, which can be embedded in HTML or incorporated into a Reports Developer report.

Oracle Reports Server is an application server that is optimized to deploy Oracle Reports Developer applications (Reports and Graphics) in a multi-tiered environment. It takes advantage of the ease and accessibility of the Web, elevating it from a static information-publishing mechanism to an environment capable of supporting complex applications.

For more information about the Forms Server and Graphics Server, refer to the Oracle Forms Developer: Deploying Forms Applications to the Web with Oracle Forms Server manual for more information.
In today’s fast-moving, competitive business world, clear and up-to-date information is needed for the accurate, expedient decision making requirements of an often geographically distributed workforce. The timely distribution of that information must be reliable, cost effective, and accessible to everyone who requires it. Oracle Reports Developer provides an unbounded, easy-to-use, scalable, and manageable solution for high-quality database publishing and reporting.

Oracle Reports Developer is a powerful Enterprise Reporting tool used by IS developers to create sophisticated dynamic reports for the Web and across the enterprise.

Oracle Reports Developer’s Server-based architecture means report consumers require only a Web browser to view reports in industry standard formats. The Oracle Reports Server supports on-demand delivery of high-quality reports over the Web through native generation of HTML with Cascading Style Sheets and Adobe’s Portable Document Format (PDF). Maintenance overheads are cut as reports are administered and maintained centrally and there is no requirement to install complex software on every user’s PC.

2.1 The Oracle Reports Server

The Reports Server enables you to implement a multi-tiered architecture for running your reports. With the Reports Server, you can run reports on a remote application server.

When used in conjunction with the Reports Web CGI or Reports Servlet, the Reports Server also enables you to run reports from a Web browser using standard URL syntax. The Reports 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.

The Reports Server keeps track of a predefined maximum number of past jobs. Information on when the jobs are queued, started, and finished is kept, as well as the final status of the report. This information can be retrieved and reviewed on Windows from the Reports Queue Manager (RWRQM60) or via the API. The Reports Queue Manager may reside on the same machine as the Reports Server or on a client machine. On UNIX, you can use the Reports Queue Viewer (RWRQV60) to view the Reports Server’s queue.

2.2 Reports Server Architecture

The Reports Server can be configured in a number of ways depending upon your requirements. When used in a Web environment, the Reports Server architecture consists of four tiers:

- The thin client tier
- The Web server tier
- The Reports Server tier
- The database tier

The range of possible configurations runs from having all of these tiers on one machine to having each of these tiers on a separate machine. The most common configurations typically have the tiers spread across three or four machines. The graphics that follow provide a conceptual view of these common configurations. 

Note: In the non-Web case, which will be discussed later, there are only three tiers because the Web server tier is not necessary.

---

1 The term tier refers to the logical location of the components that comprise the Reports Server architecture. Each of the tiers, though, could reside on the same or different machines.
2.2.1 Web architecture: server configurations

The diagrams that follow illustrate two of the most common configurations for the Reports Server in a Web environment. The key difference between the two configurations is whether the Reports Server and Web server tiers are on the same or different machines. In the first case, the Web server and Reports Server reside on the same machine. In the second case, they are on different machines. The latter case requires a slightly different setup from the first.

*Figure 2–1  Web architecture, three machine configuration*¹

¹ The Reports Web Cartridge is another way to communicate with Reports Server when you are using the Oracle Application Server as the Web server.
2.2.1.1 Processing Web reports

1. The client requests the report from their Web browser either by typing a URL or clicking a hyperlink. The Web browser passes the URL to the Web server.

2. To handle the request, the Web server invokes either the Reports Web CGI or Reports Servlet, depending upon which one you have configured.

3. The Reports Web CGI or Servlet parses the request. If necessary, users are prompted to log on. The Reports CGI or Servlet converts the request to a command line that can be executed by the Reports Server and submits it to the specified Reports Server.

1 The Reports Web Cartridge is another way to communicate with Reports Server when you are using the Oracle Application Server as the Web server.
4. If the request includes a time tolerance\(^1\), the Reports Server will check its output cache to determine whether it already has output that will satisfy the request. If it finds acceptable output in its cache, it will immediately return that output rather than executing the report.

5. The Reports Server receives the job request and queues it. When one of its runtime engines becomes available,\(^2\) it sends the command line to that runtime engine for execution.

6. The runtime engine executes the report.

7. The Reports Web CGI or Servlet receives the report output from the Reports Server and sends it to the Web server.

8. The Web server sends the report output to the client’s Web browser.

### 2.2.2 Non-Web architecture: server configuration

The non-Web architecture differs from the Web architecture in that there is no Web browser or Web server. Report requests are sent to the Reports Server from a thin client such as the Reports Launcher or command line, RWCLI.\(^6\) The non-Web architecture is useful to those who cannot use the Web to deploy their reports for some reason.

---

\(^1\) For any job request that you send to the Reports Server, you can include a TOLERANCE argument. TOLERANCE defines the oldest output that the requester would consider acceptable. For example, if the requester specified 5 minutes as the TOLERANCE, the Reports Server would check its cache for duplicate report output that had been generated within the last five minutes.

\(^2\) When you configure the Reports Server, you can specify the maximum number of runtime engines it can use. If the Reports Server is under this maximum, it may start new runtime engines to handle requests. Otherwise, the request must wait until one of the current runtime engines completes its current job.
2.2.2.1 Processing reports

1. The client requests the report using the command line (RWCLI60), the Reports Queue Manager, or the Reports Launcher (ActiveX control). If necessary, users are prompted to log on.

2. The Reports Server receives the job request and queues it. When one of its runtime engines becomes available, it sends the request to that runtime engine for execution.

3. The runtime engine executes the report.

4. The Reports Server is notified that the job has been completed.

5. If the Reports Server was called synchronously, it signals the client that the job has been completed. Note also that if the destination type (DESTYPE) for the command line client is set to localfile in the job request, the output is transferred to the client.
2.3 Reports Server Configuration Choices

The configuration of the Reports Server can vary widely depending upon the requirements of your system. Before attempting to configure the Reports Server, you must make a number of important decisions based upon your requirements. By making these decisions beforehand, you can greatly simplify the configuration process. These decisions are discussed in the following sections.

2.3.1 Enable Web and non-Web requests

As you saw in Section 2.2, "Reports Server Architecture", the Reports Server can accept job requests from both Web and non-Web thin clients. In the Web case, users run reports by clicking or typing a URL in their Web browser and, depending on the URL, the report output is served back to them in their browser or sent to a specified destination (for example, a printer). In the non-Web case, users launch job requests using client software installed on their machines (that is, Net8 and the Reports Thin Client, which is comprised of the Reports Launcher, the Reports Queue Manager, and RWCLI60).

To enable users to launch reports from a Web client, you need to install either the Reports Web CGI or Servlet with your Web server to communicate between the Web server and the Reports Server. The Web CGI or Servlet is required for your Web server to process report requests from Web clients. For more information, refer to the Section 2.3.2, "Choose the Reports Web CGI or Servlet". To enable users to launch reports from a non-Web client, you need to install the required client software (that is, Net8 and the Reports Thin Client) on each machine from which you plan to launch report requests.

From the perspective of configuration, the key differences between enabling Web and non-Web requests is as follows:

- Enabling Web requests requires that you install some additional software with your Web server, namely the Reports Web CGI or Servlet, but obviates the need to install any client software beyond a Web browser.
- Enabling non-Web requests requires that you install and maintain client software on each machine from which you want to send job requests to the Reports Server.

The Web case is clearly the most cost effective because it reduces client maintenance costs, but there may be cases where launching non-Web requests is a necessity for other reasons. The Reports Server supports both Web and non-Web requests and they are not mutually exclusive.
2.3.2 Choose the Reports Web CGI or Servlet

As discussed in Section 2.3.1, “Enable Web and non-Web requests”, to use the Reports Server in a Web environment, you must install and configure the Reports Web CGI or Servlet to handle the transmission of job requests and output between your Web server and the Reports Server. The key consideration in this choice is the following:

- If you are using a CGI-aware Web server (for example, Oracle Application Server, WebDB listener), then choose the Reports Web CGI.
- If you are using a Java-based Web server, then choose the Reports Servlet.

2.3.3 Choose the location of the Reports Server

As described in the Section 2.2, “Reports Server Architecture”, you can place the Reports Server on the same machine as your Web server or on a different machine. As you make this decision, you should consider the following:

- Having the Reports Server and the Web server on the same machine, of course, requires more of the machine’s resources. If you plan to have both on the same machine, you need to take that into account when determining the machine’s resource requirements (that is, memory and disk space).
- Having the Reports Server and the Web server on the same machine reduces network traffic. The Reports CGI or Servlet must reside on the same machine as the Web server. If the Reports Server is on a different machine, its transmissions to the Reports CGI and Servlet must travel across a network. If it is on the same machine, the transmissions do not have to travel across the network.

Once you have made your configuration choices, you are ready to configure the Reports Server. Section 3.2, “Configuring the Reports Server: Reports Web CGI Example” provides a step-by-step configuration example using Web CGI.

Section 3.4, "Reports Servlet Configuration Guidelines” provides for guidelines configuring the Reports Server using the Reports Servlet.
This chapter shows you how to configure the Oracle Reports Server, providing general instructions and then presenting an example Reports Server configuration using Web CGI. In this example, users can dynamically run reports from and receive the report output in their Web browser. No client software is necessary for users to run reports.

### 3.1 Installation and Configuration Overview

You can choose to install and configure the Reports Server automatically or manually using the step-by-step instructions found in this chapter. To do this, you must have the Oracle Report Server software.

You can choose to install and configure the Reports Server using the following methods:

- **Automatically install and configure the Reports Server.** The automatic install gets you up and running as quickly as possible performing most of the configuration for you.

- **Manually install and configure the Reports Server.** You can do a manual install when you are experienced with the Reports Server configurations and want to take full control over the configuration process. See Section 3.1.2, "Configuring the Reports Server manually" for more information.
3.1.1 Configuring the Reports Server automatically

The automatic install analyzes the dependencies on your machine and then configures the Reports Server based on the options you choose.

3.1.1.1 Step 1: Starting the Oracle Installer
To start the automatic install, start the Oracle Installer.

A configuration instructions file is produced at the end of the automatic install that summarizes the configuration settings that were created for you. This configuration instructions file informs you of any additional configuration settings that you might need to define manually.

3.1.1.2 Step 2: Choosing the Oracle Installation Settings
The first screen you see is the Oracle Installation Settings screen. On this screen you provide the following settings for Oracle Home:

- The Name field is set to DEFAULT_HOME. You do not need to change this field.
- The Location field is set by you. For example, you might use something like: D:\orant
  This field is already filled in for you if you have previously installed the Reports Server.
- The Language field is set to English. You can change this if necessary.

3.1.1.3 Step 3: Choosing the Oracle Tools Installation Options
This screen allows you to select one of the following products. In this case, choose Oracle Reports Server.

- Oracle Forms Developer
- Oracle Forms Server
- Oracle Reports Developer
- Oracle Reports Server
3.1.1.4 Step 4: Selecting the Installation Options

This screen allows you to proceed with a Typical or Custom installation.

**Note:** You may or may not receive General Information screens as you proceed through the automatic install. Either reply Yes or No, and continue. Also, in place of the `username-pc.us.oracle.com` variable in the examples, you would substitute the information for your install.

3.1.1.4.1 Typical install

The Typical installation takes you to the through the following screens.

**Note:** If you have the Control Panel open, then you must close it before continuing the install; otherwise, the installer might not be able to copy `wdbctrl.cpl` to your system.

**Server: Web Listener**

This screen provides you with two options: Use Oracle WebDB Listener or Use another Web Listener. The default is Oracle WebDB Listener. Choosing the Oracle WebDB Listener provides you with an example setup should you later want to set up your system for another Web listener.

**Process Startup**

This screen allows you to automatically set up and create the WebDB Listener and the Oracle Reports Server. If you choose Yes, then these services are set up for you. If you choose No, then only the relevant `tnsnames.ora` entry is created. Since the subsequent screens collect information to create this entry, it appears whether you choose Yes or No.

**WebDB Listener**

This screen is where you provide the WebDB Listener name and the Host Name. For example, `username-pc.us.oracle.com` for the WebDB Listener, which is the name of the PC you are installing on and can be found on the Network Control panel. And, 80 for the WebDB Listener Port number. The port number must be the same as the port number that the database uses. The default is 80.

If you press Cancel, then this setting is skipped and dependencies analyzed. The installer continues with the Reports Server parameters screen.
Installation and Configuration Overview

**Reports Server parameters**

This screen is where you provide the Reports Server TNS Name (for example, Rep60_username-pc) and the Port number (for example, 1949). This port number must be different from the WebDB Listener port number.

If you receive an ‘insufficient disk space’ notification, then you might need to temporarily free disk space until the required amount becomes available. Only then can you confirm this notification.

**Configuration instructions for Forms & Reports Server**

This screen lets you know where the configuration file has been placed. For example, D:\orant\orainst\reports6iconfig.txt. You can either review this file now or at a later time.

**Oracle WebDB Listener**

This screen lets you know where you can find two files that allow you to make additional listener modifications. For example:

http://username-pc.us.oracle.com:80//WebDB/admin_/gateway.htm

http://username-pc.us.oracle.com:80/WebDB/admin_/listener.htm

**General Information**

This final General Information screen lets you know that your installation was successful.

To verify that you can successfully access WebDB, you open your browser and enter the following:

http://username-pc.us.oracle.com:80/dev60cgi/Rwgci60.exe

where username-pc.us.oracle.com is the host name of the Web listener installation.

**3.1.1.4.2 Custom install** If you choose Custom on the Installation Options screen, the next screen you see is the Software Asset Manager screen. This screen allows you to manually deinstall products and to install other products. After successfully exiting this screen, you continue through the same path as the Typical install starting with "Server: Web Listener".
3.1.2 Configuring the Reports Server manually

To manually install and configure the Reports Server, follow these steps:

Step 1. Installing the software.

Step 2. Configuring your Web server. Refer to your Web server documentation for additional information.

Step 3. Configuring the Reports Web CGI or Servlet.


Step 5. Running a report request.

Step 6. (Optional) Modifying the configuration.

If you want to install and configure a new or additional Reports Server or deinstall an existing Reports Server, then refer to Section 3.3, "Starting or Stopping the Reports Server from the Command Line" for more information.

See the following sections for detailed instructions on configuring the Reports Server:

- Section 3.2, "Configuring the Reports Server: Reports Web CGI Example" for detailed instructions on configuring the Reports Server using Web CGI.

- Section 3.4, "Reports Servlet Configuration Guidelines" for guidelines on configuring the Reports Server using the Reports Servlet.

3.2 Configuring the Reports Server: Reports Web CGI Example

This section provides step-by-step instructions on how to manually configure the Reports Server using the Web CGI and any CGI-aware Web server. If you are using a Java-based Web server, then refer to Section 3.4, "Reports Servlet Configuration Guidelines" for more information.

Tip: This case assumes that you will install software, configure your server settings, and save .RDF files on your machine’s C: drive.

In order to make this configuration meaningful, it is necessary to make several assumptions. For the purposes of this section, the following assumptions are made:

- You are configuring the Reports Server to enable Web requests.

- You are using the Reports Web CGI with the CGI-aware Web server of your choice.

- The Reports Server is installed on a different machine than the Web server.
3.2.1 Step 1. Installing the software

You must have the Oracle Report Server software to install and configure the Reports Server. Refer to the Getting Started manual for more information on installing the Oracle Reports Server components and the Report Builder.

**Tip:** To install just the Reports Thin Client, Reports Server, or the Report Builder, start the Oracle Installer and choose the Custom Installation option. From the Available Products list box, expand the Oracle Developer - Reports node and choose Reports Thin Client, Oracle Reports Server, or Report Builder as desired. Proceed with the installation as described in the Getting Started manual.

3.2.1.1 On the Web server machine

Install the Oracle Reports Thin Client component after installing your Web server. Refer to your vendor Web server documentation for more information on installing the Web server.

3.2.1.2 On the Reports Server machine

1. Install the Oracle Reports Server component. If you want to build reports on the same machine, then install the Report Builder. The Report Builder is required only if you need to build reports. If the Reports Server machine is used only as a deployment machine, then do not install the Report Builder component.

2. When you install the Oracle Reports Server, you are asked to identify the Reports Server TNS name and Port number. This information is added to the tnsnames.ora file (located in the ORACLE_HOME\NET8\ADMIN directory).

**Tip:** You can edit the Reports Server service entry by opening the tnsnames.ora file in a text editor.

- Reports Server TNS Service Name. Type repserver.world or keep the default. where repserver.world is the name of the server instance and .world is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.

- Port. Type the port number to which the server is to listen or keep the default. The default is 1949
3. Make note of the Reports Server TNS name and port number. You will need this information to add the service entry for this Reports Server to the `tnsnames.ora` file located on the Web server machine. See Section 3.2.3.2, "Creating a service entry for the Reports Server" for more information.

### 3.2.2 Step 2. Configuring the Web server

This step is performed on the Web server machine.

**Note:** The Web `cgi-bin` directory on your Web server contains CGI executables.

1. Start your Web server.
2. Start your browser.
3. Create a listener.
4. Configure your Web server mapping and note the following physical and virtual directories:

<table>
<thead>
<tr>
<th>Directory Description</th>
<th>Physical Directory example</th>
<th>Virtual Directory Example</th>
<th>Permissions Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web <code>cgi-bin</code></td>
<td><code>c:\orant\oas\bin</code></td>
<td><code>/CGI-BIN</code></td>
<td>execute</td>
</tr>
</tbody>
</table>

The physical directory depends on directory settings chosen during the installation of your Web server software.

Refer to your vendor’s Web server documentation for more information on configuring your Web server.

### 3.2.3 Step 3. Configuring the Reports Web CGI

**Note:** If you are configuring the Reports Server using the Reports Servlet, then refer to Section 3.4, "Reports Servlet Configuration Guidelines" for more information.

This step is performed on the Web server machine.
3.2.3.1 Configuring Reports Web CGI

1. Copy \r\wcgi60.exe (located in the ORACLE_HOME\BIN directory) to your cgi-bin directory.

   Tip: The cgi-bin directory is defined in your Web server configuration. Refer to your Web server documentation for more information. In Table 3–1, "Web cgi-bin physical and virtual directories" the Web CGI physical directory is C:\your_webserver\bin.

This step is performed on your Web server machine.

3.2.3.2 Creating a service entry for the Reports Server

If the Web server is on a different machine than your Reports Server, then you must add the Reports Server service entry. This service entry was created on the Reports Service machine to the tnsnames.ora file. The tnsnames.ora file is located on your Web server machine. This enables the Web CGI executable to communicate with the Reports Server.

   Tip: If you do not remember the service entry settings for the Reports Server, then open the tnsnames.ora file located in the ORACLE_HOME\NET80\ADMIN directory on your Reports Server machine. Copy or make note of the service entry.

1. On your Web Server machine, open the tnsnames.ora file (located in the ORACLE_HOME\NET80\ADMIN directory) in a text editor.

2. Add the following Reports Server service entry:

   repserver.world=(ADDRESS = (PROTOCOL = TCP)(Host = repserver_machine.mydomain)(Port = 1949))

   where:

   repserver.world is the name of the server instance and .world is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.

   repserver_machine.mydomain is the host name or IP address of the machine.

   1949 is the port number to which the server is listening.
3.2.3.3 (Optional) Setting the default Reports Server

3.2.3.3.1 For Windows NT For Windows NT, perform the following steps.

1. On your desktop, navigate to Start→Run.
2. Type regedit to have the Registry Editor displayed.
3. From the menu, expand Hkey_Local_machine→Software→Oracle.
4. First choose Edit→New→String Value to add the following registry entry (see Section 3.2.4.1.1, "(Optional) Setting the source path"):

   REPORTS60_REPORTS_SERVER

   Then double click on the REPORTS60_REPORTS_SERVER to enter the repserver value, where repserver is the name of the Reports Server that you are configuring (the TNS service entry name of the Reports Server).

   Refer to Section 3.2.1, "Step 1. Installing the software" for additional information.

   Note: Refer to your operating system’s documentation to determine if the values you set are treated as case-sensitive.

3.2.3.3.2 For UNIX For UNIX, perform the following step.

Tip: You might want to create a shell script that sets environment variables on your Web server machine. To do this, create a file that contains the command described below.

Set the REPORTS60_REPORTS_SERVER environment variable to the name of the Reports Server:

    setenv REPORTS60_REPORTS_SERVER repserver

where repserver is the name of the Reports Server that you are configuring (the TNS service entry name of the Reports Server). Refer to Section 3.2.1, "Step 1. Installing the software" for additional information.

Tip: Refer to your operating system’s documentation to determine if the values you set are treated as case-sensitive.
3.2.4 Step 4. Configuring the Reports Server

This section discusses how you can configure the Reports Server by setting environment variables, setting the default printer, and starting the server.

3.2.4.1 Configuring the Reports Server in Windows NT

There are two primary steps for configuring the Reports Server in Windows NT:

- (Optional) setting the source path
- Starting the Reports Server

3.2.4.1.1 (Optional) Setting the source path

Alternatively, after the Reports Server is installed, you can set the source path in the Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, “Reports Server Configuration Parameters” for more information.

1. Create the source directory if it does not already exist.
2. On your desktop, choose Start → Run.
3. Type regedit to display the Registry Editor.
4. Expand Hkey_Local_machine → Software → Oracle.
5. Double-click REPORTS60_PATH.
6. Add a new reports source path (for example, C: \WEB_REPORTS).

Note: Refer to your operating system’s documentation to determine if the values you set are treated as case-sensitive.

3.2.4.1.2 Starting the Reports Server

Proceed with the following steps:

1. On the Reports Server machine desktop, choose Start → Settings → Control Panel and double-click (Services) on the Control Panel.
2. In the Services dialog box, choose Oracle Reports Server [repserver] and click Startup, which gives you the Services dialog window.
3. From the startup dialog, select This Account in the Log On As section and select an operating system user name and password. This specifies that the server is run as that user.

Tip: If you want to output to PostScript or to a printer, then ensure the user running the Reports Server service has access to a default printer. Typically, the System Account does not have access to printers.
4. Set the **Startup Type** of the service to Automatic when the system is started.

5. Click **OK**.

6. Click **Start**. A Service Control message box indicates when your Reports Server has started. If your Reports Server cannot start, then refer to Appendix F, "Troubleshooting" for more information.

   **Tip:** When you start the Reports Server for the first time, a Reports Server configuration file (for example, repserver.ora) is created in the ORACLE_HOME\REPORT60\SERVER directory. The setting for your Reports Server cache is set by default. You can change the cache directory, or set the report's source path by modifying the configuration file. If you modify the configuration file, then stop and restart the Reports Server for the changes to take effect.

### 3.2.4.2 Configuring the Reports Server on UNIX

There are two primary steps for configuring the Reports Server on UNIX:

- (Optional) Setting environment variables
- Starting the Reports Server

#### 3.2.4.2.1 (Optional) Setting environment variables

   **Tip:** You might want to create a shell script that sets the environment variables and launches the RWMTS60 executable. To do this, create a file that contains the commands described below.

   **Note:** The executable names are lowercase in the UNIX environment. The UNIX environment is case-sensitive.

1. Create a directory for your source reports (for example, /WEB_REPORTS).
2. Set the REPORTS60_PATH environment variable to locate the reports:

   ```
   setenv REPORTS60_PATH $WEB_REPORTS
   ```

   The REPORTS60_PATH should include all directories that you want Oracle Reports Server to look for (for example, Reports source files, libraries).

   **Tip:** Alternatively, after the Reports Server is installed, you can set the source path in the Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, "Reports Server Configuration Parameters" for more information.
3. Set the TNS_ADMIN environment variable to point to the location of the tnsnames.ora file:

   setenv TNS_ADMIN $ORACLE_HOME/NET80/ADMIN

   **Tip:** Refer to your operating system’s documentation to determine if the values you set are treated as case-sensitive.

### 3.2.4.2.2 Starting the Reports Server

During the installation of the Oracle Reports Server, the RWMTS60 command that starts the Reports Server was added to the shell script.

1. Run the shell script to start the Reports Server.

   **Tip:** When you start the Reports Server for the first time, a Reports Server configuration file (for example, repserver.ora) is created in the $ORACLE_HOME/REPORT60/SERVER directory. The setting for your Reports Server cache is set by default. If you want to change the cache directory, or if you want set the report’s source path, then you can do so by modifying the configuration file. If you modify the configuration file, then you need to stop and restart the Reports Server to acknowledge the changes.

2. (Optional) From the $ORACLE_HOME/BIN directory, run the following command line to ensure that the Reports Server is running:

   ```
   rwrqv60 server=repserver
   ```

   Status columns (for example, NAME, OWNER, DEST) for this Reports Server are displayed. Currently, though, no status information is available since no jobs are running.

   **Tip:** To stop the Reports Server, press ctrl-C.

   **Tip:** If you want to output to PostScript or to a printer, the printer must be configured in the uiprint.txt file (located in the $ORACLE_HOME/GUICOM6/TK60/ADMIN directory).
3.2.5 Step 5. Running a report request from a Web browser

1. Ensure the Reports Server is configured properly. In a Web browser, make the following request:

   http://your_webserver/cgi-bin/rwcgi60.exe?report=your_report.rdf+
   userid=username/password@my_db+desformat=html+destype=cache

   where username/password@my_db is replaced with a valid database logon. Replace my_db with the tnsnames.ora entry you created for earlier for the Reports Server (Section 3.2.3.2, "Creating a service entry for the Reports Server").

   Notice that the SERVER command line argument is missing from the request. It is not required if you set the REPORTS60_REPORTS_SERVER environment variable on your Web Server machine.

   Tip: If the report does not run or if you receive an error message, then refer to Appendix F, "Troubleshooting" for more information.

2. (Optional) View the status of the request.

   ■ For Windows NT, start the Reports Queue Manager, choose to view the repserver queue. See the Reports Queue Manager online help for more information.

   ■ For UNIX run the following command:

   rwrqv60 server=repserver showjobs=current

   This completes the Reports Server configuration example using Web CGI for CGI-aware Web servers. You have successfully configured the Reports Server for dynamic reporting.

3.2.5.1 Next steps

1. (Optional) Tune the Reports Server to optimize performance or implement additional features, such as access control. Refer to Section 3.2.6, "Step 6. (Optional) Modifying the Reports Server configuration" for more information. Doing this step eliminates the need to show all of the parameters as shown in Section 3.2.5, "Step 5. Running a report request from a Web browser"; thus protecting your user name and password information.

2. Make reports available to users. See Chapter 4, "Running Report Requests" for more information on how to specify run requests and make them available to users.
3.2.6 Step 6. (Optional) Modifying the Reports Server configuration

When you start the Reports Server for the first time, the Reports Server is set with default configuration settings (for example, maximum and minimum engines). At some point, you might want to modify the Reports Server configuration to tune performance, set up monitoring controls, or implement additional features.

- To update the database with job queue information, refer to Section 3.2.6.1, "Updating the database with job queue activity" for more information.
- To control user access to reports, refer to Chapter 5, "Controlling User Access to Reports" for more information.
- To configure the Reports Server for load balancing, refer to Chapter 6, "Configuring Reports Server Clusters" for more information.
- To modify a report at runtime based on the audience, refer to Chapter 7, "Customizing Reports at Runtime" for more information.
- To modify the Reports Server configuration file, see Appendix B, "Reports Server Configuration Parameters" for more information about the available configuration parameters.
- Modify environment variables, refer to Appendix C, "Environment Variables" for more information.

3.2.6.1 Updating the database with job queue activity

You can set up your database to take snapshots of the Reports Server queue activity whenever jobs are run. When you start the Reports Server, a connection to the database is made. By default, the Reports Server calls an API to delete queue information when the server restarts and to update the queue information in the database table.

You can edit the source for the API in the `rw_server.sql` script to override the defaults (for example, to prevent the queue from being deleted when restarting the Reports Server). The prototype of the procedure (the procedure name and the parameters it expects) should not be edited.

To connect to the database, you set the REPOSITORYCONN parameter in the configuration file. To create the queue activity table in the database, you run the `rw_server.sql` script.

If you change the contents of the script, then you have to run it as that user, and then restart the Reports Server for the changes to take effect.
3.2.6.1.1 On the Reports Server machine:

1. Open the `repserver.ora` configuration file (located on the `ORACLE_HOME\REPORT60\SERVER` directory) in a text editor.

2. Add the following configuration parameter:

   ```
   REPOSITORYCONN="repserver_schema/password@my_db"
   ```

   where `repserver_schema/password@my_db` is the connection string to the schema in your database that takes snapshots of queue activity of the specified Reports Server. In this case, `repserver_schema` is the schema for `repserver` queue activity.

   Tip: If you want to take snapshots of queue activity from many Reports Servers, then it is recommended that you create a different schema in your database for each Reports Server that requires snapshots. This prevents you from losing queue activity data when you restart the Reports Server.

3. The `repserver_schema` must have, at a minimum, create table and create package privileges to run the `rw_server.sql` script (which is located in the `ORACLE_HOME\REPORT60\SQL` directory) from the command line. At the command line prompt, type:

   ```
   cd C:\ORACLE_HOME\REPORT60\SQL <RETURN>
   plus80 username/password@my_db <RETURN>
   @rw_server.sql <RETURN>
   quit <RETURN>
   ```

4. Stop and restart the Reports Server to accept the changes made to the configuration file. When the Reports Server starts up, it connects to the database.

   **Note:** When you restart your Reports Server, queue activity in the database is deleted by default. You can override the default by editing the API.
3.3 Starting or Stopping the Reports Server from the Command Line

You can start additional Reports Servers, or stop existing Reports Servers from the command line. You must have already installed the Reports Server component to do this. See Section 3.1, "Installation and Configuration Overview" for more information.

RWMTS60 is the executable for the Reports Server. For Windows NT, three commands are associated with it:

- **Install**, enables you to install the Reports Server as a service on Windows NT.
- **Listen**, enables you to install the Reports Server on Windows NT as a non-service.
- **Uninstall**, enables you to deinstall the Reports Server on Windows NT as a service.

You can install, listen, and deinstall the Reports Server manually or in batch mode. Set the BATCH argument to Yes to ignore any error messages raised during the installation and allow the operation to complete without interruption. Any errors or status information raised during installation can be viewed in the Reports Server log file (located on `ORACLE_HOME\REPORT60\SERVER\repservr.log`). The default for BATCH is No.

You can:

- Start another Reports Server. Refer to Section 3.3.1, "Starting another Reports Server" for more information.
- Start the Reports Server in Windows NT as a non-service. Refer to Section 3.3.2, "Starting the Reports Server on Windows NT as a non-service" for more information.
- Stop the Reports Server. Refer to Section 3.3.3, "Stopping the Reports Server" for more information.

3.3.1 Starting another Reports Server

To start another Reports Server, you must:

- Create a service entry for the Reports Server.
- (Optionally) Set environment variables (for example, set the source path).
- Start the Reports Server.
3.3.1.1 Creating a service entry for the Reports Server

1. On your Reports Server machine, open the tnsnames.ora file (located in the ORACLE_HOME\NET80\ADMIN directory) in a text editor.

2. Add the following Reports Server service entry:

repserver.world = (ADDRESS = (PROTOCOL = TCP) (Host = repserver2.machine.mydomain) (Port = 1950))

where:

repserver2.world is the name of the server instance and .world is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.

repserver.machine.mydomain is the host name or IP address of the machine.

1950 is the port number to which the additional server is listening. The initial server is set to listen to 1949 as the default.

3. Repeat steps 1 and 2 on the Web server machine if it is installed on a different machine than the Reports Server.

3.3.1.2 Configuring the Reports Server on Windows NT

The following steps are necessary to configure the Reports Server on Windows NT:

■ (Optional) Setting the source path
■ Starting the Reports Server

3.3.1.2.1 (Optional) Setting the source path

In order for the Reports Server to run the requested report, it needs to know where the report is located. You can set the report’s source path in the REPORTS60_PATH environment variable.

Alternatively, after the Reports Server is installed, you can set the source path in the Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, "Reports Server Configuration Parameters" for more information.

1. Create the source directory if it does not already exist.
2. On your desktop, choose Start→Run.
3. Type regedit to display the Registry Editor.
4. Expand Hkey_Local_machine → Software → Oracle.
5. Double-click REPORTS60_PATH.
6. Add a new reports source path (for example, C:\WEB_REPORTS).
   Note: Refer to your operating system’s documentation to determine if the values you set are case-sensitive.

3.3.1.2.2 Starting the Reports Server
2. Type the following command line argument:
   rwmts60 -install repserver tcpip

   or for batch mode:
   rwmts60 -install repserver batch=yes

   Tip: Repserver does not need to have the domain qualifier (for example, .world) appended to it.
3. Click Yes to install the Reports Server as an NT Service and Click OK to start up the service.
4. On your desktop, choose Start → Settings → Control Panel and double-click (Services).
5. In the Services dialog box, choose Oracle Reports Server [repserver] and click Startup.
6. From the startup dialog, select This Account in the Log On As section and type or select an operating system user name and password. This specifies that the server is run as that user.
   Tip: If you want to output to PostScript or to a printer, then ensure the user running the Reports Server service has access to a default printer. Typically, the System Account does not have access to printers.
7. Set the service to start automatically when the system is started.
8. Click OK.
9. Click Start. A Service Control message box indicates when your Reports Server has started. If your Reports Server cannot start, then refer to Appendix F, "Troubleshooting" for more information.

Tip: When you start the Reports Server for the first time, a Reports Server configuration file (for example, repserver.ora) is created in the ORACLE_HOME\REPORT60\SERVER directory. The setting for your Reports Server cache is set by default. You can change the cache directory, or set the report’s source path by modifying the configuration file. If you modify the configuration file, then stop and restart the Reports Server for the changes to take effect.

3.3.1.3 Configuring the Reports Server on UNIX
The following steps are necessary to configure the Reports Server on Windows NT:

■ (Optional) Setting the environment variables
■ Starting the Reports Server

3.3.1.3.1 (Optional) Setting the environment variables
You can set two optional environment variables. The first lets the Reports Server know where the requested report is located. You can set the report’s source path in the REPORTS60_PATH environment variable. The second sets the location of the tnsames.ora file.

Tip: You might want to create a shell script that sets the environment variables and launches the RWMTS60 executable. To do this, create a file that contains the commands described below.

1. Create a directory for your source reports (for example, /WEB_REPORTS).
2. Set the REPORTS60_PATH environment variable to locate the reports:

   setenv REPORTS60_PATH $WEB_REPORTS

   Tip: Alternatively, after the Reports Server is installed, you can set the source path in the Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, "Reports Server Configuration Parameters" for more information.
3. Set the TNS_ADMIN environment variable to point to the location of the tnsnames.ora file:

    setenv TNS_ADMIN $ORACLE_HOME/NET80/ADMIN

**Tip:** Refer to your operating system’s documentation to determine if the values you set are case-sensitive.

### 3.3.1.3.2 Starting the Reports Server

1. From the $ORACLE_HOME/BIN directory, run the following command line to run the Reports Server in the foreground:

    rwmts60 name=repserver

   or to run the Reports Server in the background:

    rwmts60 name=repserver &

2. To determine if your server is running, go to step 2. If your Reports Server does not start or you get an error message, then refer to Appendix F, "Troubleshooting" for more information.

   **Tip:** When you start the Reports Server for the first time, a Reports Server configuration file (for example, repserver.ora) is created in the $ORACLE_HOME/REPORT60/SERVER directory. The setting for your Reports Server cache is set by default. If you want to change the cache directory, or if you want set the report’s source path, then you can do so by modifying the configuration file. If you modify the configuration file, then you need to stop and restart the Reports Server to acknowledge the changes.

3. (Optional) From the $ORACLE_HOME/BIN directory, run the following command line to ensure that the Reports Server is running:

    rwrqv60 server=repserver

   Status columns (for example, NAME, OWNER, DEST) for this Reports Server are displayed. Currently, though, no status information is available since no jobs are running.

   **Tip:** If you want to output to PostScript or to a printer, then the printer must be configured in the uiprint.txt file (which is located in the $ORACLE_HOME/GUICOM6/TK60/ADMIN directory).
3.3.2 Starting the Reports Server on Windows NT as a non-service

Run the following command line:

```bash
rwmts60 -listen repserver
```

or in batch mode:

```bash
rwmts60 -listen repserver batch=yes
```

Tip: Repserver does not need to have the domain qualifier (for example, .world) appended to it.

3.3.3 Stopping the Reports Server

3.3.3.1 On Windows NT

2. Type the following command line argument:

```bash
rwmts60 -uninstall repserver
```

or in batch mode:

```bash
rwmts60 -uninstall repserver batch=yes
```

Tip: Repserver does not need to have the domain qualifier (for example, .world) appended to it.

3.3.3.2 On UNIX

Ensure that focus is in the correct window and press ctrl-C.

3.4 Reports Servlet Configuration Guidelines

You can configure the Reports Server using the Reports Servlet if you are using a Java-based Web server (for example, Oracle Lite, Oracle Application Server). The Reports Server configuration using the Reports Servlet is similar to the Web CGI configuration. However, you must also install servlet-enabled Web server software.
Use the following guidelines to configure your Java-based Web server using the Reports Servlet.

1. Install the Oracle Reports Thin client software and servlet-enabled Web server software. Refer to the Getting Started manual for more information on installing the Oracle Reports Thin client. Refer to your Web server documentation for more information on installing the Web server.

2. Configure your Web server. Refer to the Web server’s documentation for more information.

3. Configure the Reports Servlet. Refer to Section 3.4.1, "Configuring the Web server" for more information.


### 3.4.1 Configuring the Web server

This step is performed on the Web server machine by doing the following:

1. Add the following entry to the servlet properties file (for example, `servlet.properties`)
   ```
   # Oracle Report servlet
   servlet.RWServlet.code=oracle.reports.rwcgi.RWServlet
   ```

2. Set the CLASS_PATH environment variable to directory containing the Oracle Reports RWServlet.class file.

3. When you complete the Reports Server configuration, ensure the Reports Server is configured properly. In a Web browser, make the following request:
   ```
   http://your_webserver/RWServlet?report=your_report.rdf+
   userid=username/password@m_db+desformat=html+destype=cache
   ```

   where `username/password@m_db` is replaced with a valid database logon. Replace `my_db` with the `tnsnames.ora` entry you created for earlier for the Reports Server (Section 3.2.3.2, "Creating a service entry for the Reports Server").

   Notice that the SERVER command line argument is missing from the request. It is not required if you set the REPORTS60_REPORTS_SERVER environment variable on your Web Server machine to the repserver.
This chapter discusses various ways to specify report requests. The following topics are covered:

- Report request methods
- Duplicate job detection
- Using a mapping file to simplify run requests
- Specifying URL run requests
- Scheduling reports requests to run automatically

## 4.1 Report Request Methods

You can run report requests using various request methods. They are listed below:

- **RWCLI60 command line** enables you to run a report request from the command line prompt. RWCLI60 is an executable that parses and transfers the command line to the specified Reports Server. It uses a command line similar to the Reports Runtime executable (RWRUN60). An RWCLI60 command line request is made using a non-Web architecture. A typical command line request looks like the following:

```
RWCLI60 REPORT=my_report.rdf USERID=username/password@my_db
SERVER=repserver DESTYPE=HTML DESFORMAT=cache
```

See Appendix A, "RWCLI60 Command Line Arguments” for a list of valid RWCLI60 command line arguments.
■ **URL Syntax** enables you to run a report request from a Web browser. Web CGI, and Servlet converts the URL syntax into an RWCLI60 command line request that is processed by the Reports Server. When the report has finished processing, the output is sent to an HTML or PDF file in a location known to the Web server, which is served back to the requesting Web browser. You can provide users the URL syntax needed to make the report request from their browser, or you can add the URL syntax to a Web site as a hyperlink. The remainder of this chapter discusses this method in more detail.

■ **WebDB component** enables you add a link as an Oracle WebDB component to a WebDB site. This link points to a packaged procedure that contains information about the report request. Reports Developer system administrators use Oracle WebDB wizards to create the packaged procedure making it more convenient and secure to publish the report via the Web. Authorized users accessing the WebDB site simply click the link to run the report. System administrators can run the report directly from the wizard. See Chapter 5, "Controlling User Access to Reports" for more information.

■ **ActiveX control** exposes the Reports Server through industry-standard ActiveX technology enabling you to run reports from any ActiveX container. The Reports Launcher is an example of an ActiveX container. Refer to the ActiveX and Reports Launcher online help for more information.

■ **SRW.RUN_REPORT** is a packaged PL/SQL procedure that executes a Reports Runtime command. When you specify the SRW.RUN_REPORT command line, set the SERVER argument to the Reports Server TNS service entry name to cause the SRW.RUN.REPORT command to behave as though you executed an RWCLI60 command. Refer to the Report Builder online help for more information.

### 4.2 Duplicate Job Detection

When you run a report with the DESTYPE set to cache or the TOLERANCE set to any number minutes (that is, 0 or greater), a copy of the report output is saved in the Reports Server’s cache. Subsequently, if an identical report is run (that is, with the exact command line arguments), then the current request is recognized as a duplicate job. The Reports Server will reuse the output from the cache instead of executing the report again if it is requested within the specified tolerance (for example, TOLERANCE=10). When the prior job is finished, or if it has already finished, the cached output will be used for the subsequent report, too. If one of the jobs is canceled (for example, canceled from the Reports Queue Manager), the runtime engine will run the other report normally.
Refer to Appendix A, "RWCLI60 Command Line Arguments" for more information about the DESTYPE and TOLERANCE command line arguments.

Usage notes

- The following command line arguments are compared to detect duplicate jobs: REPORT, USERID, DESFORMAT, paramform, currency, thousands, decimal, pagesize, orientation, mode, and all user parameters.

- To distribute the output of a report to multiple destinations, you can run the report once on a server, and then submit the same command to the same server with a different destination and tolerance. The Reports Server will detect the duplicate job and redistribute the cached file to the new destination.

- Duplicate job detection operates independently on each instance of a repeated job.

- You can set the cache size through the Reports Queue Manager or manually by setting the CACHESIZE parameter in the Reports Server configuration file. The Reports Server will attempt to keep the total size of cache files below this limit, deleting the least recently used files from the cache first. In addition, you can empty the cache through the Reports Queue Manager.

Refer to the Reports Queue Manager online help, or see Appendix B, "Reports Server Configuration Parameters" for more information on setting the cache.

- If a report is being processed when an identical job is submitted, the Reports Server will reuse the output of the currently running job even if TOLERANCE is not specified or is equal to zero. Suppose that job_1 is currently being run by one of the Reports Server's engines and someone else submits job_2, which is identical to job_1. The Reports Server will use the output from job_1 for job_2. In this case, processing job_2 is significantly faster since job_2 is not sent to an engine for execution.

4.3 Using a Key Map File

If you choose to provide users with the URL syntax or add the URL syntax as a hyperlink to any Web site, you can use a key map file to simplify or hide parameters in your URL requests. Key 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 users (for example, the database connect string).

Restricting the parameters users can use to run a report.

**Note:** A more convenient and secure way to publish reports on a Web site is to create a WebDB component. See Chapter 5, "Controlling User Access to Reports" for more information.

A map file takes a URL parameter and maps it to the command line arguments that govern the report request. For example, one argument in the URL request syntax could map to all of the command line arguments needed to run the report. By using key mapping, the command line arguments are all hidden from the user.

Below is an example of a key mapping for a restricted run with a Parameter Form:

A submission of:

```
http://your_webserver/cgi-bin/rwcki60.exe?key+par1+par2+parN
```

where the key mapping file contains:

```
KEY: module=myreport deptno=%1 myparam=%2 %*
```

generates the equivalent of the following command line request:

```
RWCLI60 module=myreport deptno=par1 myparam=par2 parN
```

### 4.3.1 Enabling key mapping

Key mapping is enabled when either of the two following conditions are met:

- The REPORTS60_CGIMAP (Web CGI) environment variable on the Web server machine specifies the name of a valid key map file. See Appendix C, "Environment Variables" for more information.

- A valid file with the standard file name, `cgicmd.dat`, is present in `ORACLE_HOME\REPORT60` directory on the Web server machine.

**Usage notes**

- When key mapping is enabled, all RWCGI60 URLs are treated as if the first argument is a key. The key map file searches for this key. If the key is found, its defined value is substituted into the command line for the Reports Server. If it is not found, an error is generated.

- When submitting a URL via an HTML form, the key is coded as an input of type hidden.
4.3.2 Mapping URL parameters

This section describes how to add key mapping entries to a key map file.

On the Web server machine:

1. Open cgicmd.dat (Web CGI) file, located in the ORACLE_HOME\REPORT60 directory, in a text editor.

   Tip: Type: http://your_webserver/cgi-bin/rw CGI60.exe/showmap? in your Web browser to verify the mapping file that is being used.

2. Add a key mapping entry. A basic key mapping entry looks similar to the following:

   key1: REPORT=your_report.rdf USERID=user_name/password@mydb DESFORMAT=html
   SERVER=repserver DESTYPE=cache

   where key1 is the name of the key.

   Except for the special parameters that are described in the file itself, the command line arguments follow the syntax rules of RW CLI60. See Appendix A, "RW CLI60 Command Line Arguments" for more information about the RW CLI60 command line arguments.

   Tip: If you set the REPORTS60_REPORTS_SERVER environment variable and are sending the request to the default server, you can omit the SERVER command line argument. See Appendix C, "Environment Variables" for more information.

3. Add or update the hyperlinks on your Web page. See Section 4.4.2, "Specifying a report request from a Web browser".

4.4 Specifying Report Requests

4.4.1 Building a report

1. On the machine where your Reports Server is located, create the reports source directory (for example, C:\WEB_REPORTS) for saving the reports using the path. Ensure that this directory is set in the SOURCEDIR parameter in the Reports Server configuration file. See Appendix B, "Reports Server Configuration Parameters".
Tip: The reports source path can also be set in the REPORTS60_PATH environment variable. See Appendix C, "Environment Variables" for more information.

Start the Report Builder and build a report. You can save this report as an .RDF or .REP file. Be sure to copy this report definition file to the reports source directory on the Reports Server machine (for example, C:\WEB_REPORTS). Refer to the Building Reports manual or Report Builder online help for more information about building a report.

1. For online help on this task, choose Help → Report Builder Help Topics
2. On the Index page, type... report, building
3. Then click Display to view help topic... Building a standard report

2. Make this report available to users. See Section 4.4.2, "Specifying a report request from a Web browser".

4.4.2 Specifying a report request from a Web browser

You can provide the user with the URL syntax needed to make a report request, or you can add the URL syntax to a Web page as a hyperlink.

Note: A more convenient and secure way to publish reports on a Web site is to create a WebDB component. See Chapter 5, "Controlling User Access to Reports" for more information.

URL syntax can be presented in the following forms:

- Full URL request that looks similar to the following:
  
  http://your_webserver/cgi-bin/rwcgi60.exe?report=your_report.rdf
  +userid=user_name/password@mydb+server=repserver+desformat=html
  +destype=cache

  Note: If you require additional command line arguments, see Appendix A, "RWCLI60 Command Line Arguments" for a list of valid RWCLI60 command line arguments.
Specifying Report Requests

- Simplified URL request using key mapping that looks similar to the following:
  
  http://your_webserver/cgi-bin/rwcgi60.exe?report=key1

  **Tip:** If you set the REPORTS60_REPORTS_SERVER environment variable and are sending the request to the default server, you can omit the SERVER command line argument. See Appendix C, "Environment Variables" for more information.

  To add the URL syntax to a Web page as a hyperlink:

  1. Add the URL request as a hyperlink to your Web page. The syntax looks similar to the following:

     <A HREF="http://my_webserver/cgi-bin/rwcgi60.exe?key1">My report</A>

  2. Provide users the Web site URL that publishes the report request. Users click the link to run the report.

     If the report does not run or display in Web browser as expected, see Appendix F, "Troubleshooting" for more information.

4.4.3 Scheduling reports to run automatically

You can also use the server to run reports automatically from the Queue Manager or from Oracle WebDB. The scheduling feature enables you to specify a time and frequency for the report to run.

Refer to the Reports Queue Manager online help for more information about scheduling your reports.

If you publish your reports on a WebDB site as WebDB component, you can schedule these report requests to run automatically and push the resulting reports to specified folders on the site. See to Chapter 5, "Controlling User Access to Reports" for more information.
Access control enables you to restrict user access to reports that are run on Oracle Reports Servers. Reports Developer uses WebDB to perform a security check that ensures that users have the necessary privileges to run reports on restricted Reports Servers and printers. Access control determines the following:

- What report definition files, Reports Servers, and printers are restricted.
- Who has access privileges to run requested reports on a restricted Reports Server and output to a restricted printer.
- When report definition files, Reports Servers, and printers are available to run.
- How report output is delivered by restricting report request options (that is, required and optional parameters) that are available to users at runtime. This includes specifying the Reports Servers and printers that are available to users.

WebDB stores information about the report definition file (that is, how to run the report) as a packaged procedure. In order to run a report, WebDB also needs to store access control information about the restricted Reports Server that accepts the request, and any printers that are used to print report output. These access controls are added using Reports Developer Security wizards in WebDB. Only users who have Reports Developer system administrator privileges can add access controls in WebDB.
You can make report requests available to users on the Web by doing the following:

- Adding a link as a WebDB component to a WebDB site that points to the report’s packaged procedure. See Section 5.3.8, "Step 8. Make the report available to users" for more information about this method.

- Scheduling a request to run automatically and push the report output to a WebDB site for users to view. See Section 5.3.9, "Step 9. Schedule the report to run and push the output to a WebDB site" for more information.

- Adding standard URL syntax to a Web site as a hyperlink. See Section 4.4, "Specifying Report Requests" for more information.

**Note:** System administrators can run report requests from Reports Developer Security in WebDB. See Section 5.3.7, "Step 7. Set parameter values on the Reports Developer parameter form" for more information.

### 5.1 Access Control Configuration and Setup Overview

This section describes how to configure the Reports Server for access control and how to add access information in WebDB that will be used to run report requests to restricted Reports Servers.

The steps below assume that you have already configured the Reports Server using Web CGI or Servlet. See Chapter 3, "Configuring the Reports Server" for more information. See Section 5.3, "Setting Up Access Controls in WebDB" for a detailed example on implementing access control in Reports Developer.

**Installing and configuring Reports Developer security**


**Setting up access control**

Step 1. Configure the Reports Server for access control.

Step 2. Create user accounts.

Step 3. (Optional) Create availability calendars in WebDB.

Step 4. Add access to the printer in WebDB.

Step 5. Add access to the Reports Server in WebDB.

Step 6. Add access to the report definition file in WebDB and create a packaged procedure.
Note: You can batch register multiple reports in WebDB using the Reports Developer Batch Registering utility. Refer to the Reports Developer Batch Registering Reports technical white paper located on the OTN (http://technet.oracle.com).

Step 7. Set parameter values on the Parameter Form.
Step 8. Publish the report request on a WebDB site.
Step 9. (Optional) Schedule the report to run and push the output to a WebDB site.

5.2 Installing and Configuring Reports Security in WebDB

Installing and configuring the Reports Developer Security in WebDB involves installing WebDB and the Reports Server Security feature, and then setting the authentication cookie domain.

Once Reports Server Security is installed and configured in WebDB, see Section 5.3, "Setting Up Access Controls in WebDB" for information on configuring the Reports Servers for access control and adding access to Reports Servers, reports, and printers in WebDB.

5.2.1 Step 1. Configure reports security in WebDB

You must do the following to install and configure the Reports Developer Security feature in WebDB:

- Install WebDB release 2.2 or later.
- Install Reports Security feature in WebDB.
- Set the authentication cookie domain.

5.2.1.1 To install WebDB

Install WebDB into an Oracle 7.3.4, Oracle 8.0.5, or Oracle 8i database if it has not been installed already. Refer to the Oracle WebDB Getting Started-Installation and Tutorial manual for more information.
5.2.1.2 To install the Reports Server security packages in WebDB

You can install the Reports Server security packages from any machine (for example, where your WebDB or your Reports Server is installed).

1. If you want to install just the Reports Server security packages, start the Oracle Installer and choose the **Custom Installation**. From the Available Products list box, expand the Oracle Reports Developer node and choose **Reports Server Security Packages**. Refer to the **Getting Started** manual for more information about the installation process.

2. When the installation is complete, run the SQL script that will install the security packages in WebDB:
   - For Windows NT, choose **Install Reports Developer Security** from the Oracle Reports Developer Admin. menu.
   - For UNIX, go to the `ORACLE_HOME/REPORT60/SERVER/SECURITY` directory and type the following at the command line:
     ```sql
     sqlplus /nolog @rwwvins.sql
     ```

3. Type the following at the Enter Connection String prompt to log on to the WebDB schema (username/password@database).

   *Table 5–1  Connection to log on to the WebDB schema*

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>A user name with DBA privileges that logs you on to the WebDB schema. Contact your DBA if you cannot log on to the WebDB schema.</td>
</tr>
<tr>
<td>Password</td>
<td>A password that logs you on to the WebDB schema.</td>
</tr>
<tr>
<td>Database</td>
<td>The name of the database that connects you to the WebDB schema.</td>
</tr>
</tbody>
</table>

4. When the SQL script is complete, start WebDB and log on to the WebDB schema.

5. Click **Administrator** at the main menu. You should see the **Reports Developer Security** menu item.
5.2.1.3 To set the authentication cookie domain
You set the authentication cookie domain so that the cookie can send the authentication information to the Reports Server where the report is sent.

1. For WebDB online help on this task, click the help button on the title bar...
2. Click and type... authentication cookie
3. Click Find and then click... Setting the authentication cookie to display the topic.

On the machine where WebDB is installed:

1. Open the wdbsvr.cfg file in a text editor (located in the ORACLE_HOME\LISTENER\CFG directory). Under the [SERVER] section, set the configuration parameter using the following syntax:

   ORCookieDomain=my_company.com

   where my_company.com is the domain name of the Reports Server
2. Save your changes and close the configuration file.

5.3 Setting Up Access Controls in WebDB
This example provides step-by-step instructions that will help you configure your Reports Server for access control. You will add access to the report definition file, Reports Server, and printer in WebDB. Finally, you will publish the report request on a WebDB site so that authorized users can run this restricted report.

This example assumes the following:

- The Reports Server is configured using the Web CGI configuration example. See Section 3.2, "Configuring the Reports Server: Reports Web CGI Example" for more information.
- A printer that Reports Developer can recognize must be installed and running.
- The Reports Developer system administrator has WebDB site administrator privileges. This enables the Reports Developer system administrator to add items to a WebDB site and grant Manage Item privileges to other users.
You have access to the security.rdf. This report generates a 401K report for employees. Information about this report will be added in WebDB. This file is provided for you and is located in the ORACLE_HOME\TOOLS\DOC60\US\RBBR60 directory.

You must be able to access the Oracle Reports Developer demo tables to run the security.rdf file on the Reports Server. Use the demo CD that came with your product package to install the SQL scripts that are used to install the demo tables in your database. These SQL scripts can be run from the Start → Programs → Programs menu.

The 401K report that you add access to in WebDB contains vested 401K portfolio information of four fictional employees. You want to restrict access to this confidential report only to these four employees. Further, you want to ensure that the requesting employee can access only his personal information, not other employees' information. This can be achieved by doing the following:

- Restrict user access to the report itself. For the purposes of this example, you will create a user account for Jeff Abers, one of the 401K participants, and then give this user access to the report.

- Restrict authorized users access to only their personal information. Users with access to run the report will need to enter the correct last name and social security number combination to retrieve their personal 401K summary. The security.rdf report was built with two user parameters: last name and social security number (SSN). In WebDB, you will build a Runtime Parameter form that contains a list of values of the last names of 401K participants and an unrestricted parameter for the social security number.

Suppose that Jeff Abers wants to review his 401K investments. On the Runtime Parameter Form, in addition to the destination parameters, he will need to choose his last name from a list of values and then enter his social security number. When he runs the report, he must log on. WebDB checks that he has the access privileges needed to run the report. If he logs on successfully, the Reports Server processes the request. If he entered the correct last name and social security number combination, then his personal 401K report is delivered as requested.
5.3.1 Step 1. Configure the Reports Server for access control

The Reports Server must be installed and configured before you can perform this step. See Chapter 3, "Configuring the Reports Server" for information.

To configure the Reports Server for access control, you will do the following:

- Create a TNS names alias that connects to WebDB.
- Restrict access to the Reports Server.

5.3.1.1 To create the TNS names alias that connects to WebDB

You need to create a TNS names alias for WebDB in the tnsnames.ora file on the machine where the Reports Server is installed. This enables the Reports Server to communicate with WebDB.

You can create the TNS names alias using the Net8 Easy Config tool, or you can create one by editing the tnsnames.ora file in a text editor.

To create TNS names alias, you will need the following information:

- A TNS names alias for the WebDB instance.
- The host name of the database where WebDB is installed.
- The port number of the database where WebDB is installed.
- The System Identifier (SID) of the database where WebDB is installed.

**Tip:** You can find the host name, port number, and SID in the tnsnames.ora file in the ORACLE_HOME\NETWORK\ADMIN directory on the machine where the database is installed.

On the Reports Server machine:

1. Do one of the following:
   - Start Net 8 Easy Config (if it is installed on your machine) and follow the instructions on the wizard to help you create the TNS names alias.
   - Open the tnsnames.ora file located in the ORACLE_HOME\NET80\ADMIN directory. Go to step 2.

**Tip:** If you installed the Reports Server Security feature from your Reports Server machine, then a TNS names alias for WebDB has already been created for you. You can skip this step and go to Section 5.3.1.2, "To restrict access to the Reports Server".
2. Add the following TNS names alias to connect to WebDB:

```sql
sec_rep.world =
  (DESCRIPTION =
    (ADDRESS =
      (PROTOCOL = TCP)
      (HOST = my_pc.my_domain)
      (PORT = 1521)
    )
    (CONNECT_DATA = (SID = ORCL))
  )
```

where:

- `sec_rep.world` is the name of the WebDB server instance and `.world` is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit `.world` from the name of the server instance.
- `TCP` is the protocol address information.
- `my_pc.my_domain` is the host name or IP address of the machine where WebDB is installed.
- `1521` is the port number to the database where WebDB is installed.
- `ORCL` is the Oracle System Identifier for the database where WebDB is installed.

3. Save and close the tnsnames.ora file.

5.3.1.2 To restrict access to the Reports Server

To restrict access to the Reports Server, you set the SECURITYTNSNAME parameter in the Reports Server configuration file. Once set, access control is enforced. Users will be required to authenticate themselves to run report requests to this restricted Reports Server.

To run a report request, access to the report definition file must be added in WebDB. If you want to run unrestricted report requests, ensure that the Run Only Registered Report Definition Files option is unchecked in the Server Access wizard in WebDB for this Reports Server. Users, however, will still need to authenticate themselves to the Reports Server to run the report.
On the Reports Server machine:

1. Open the repserver.ora configuration file (located in the ORACLE_HOME\REPORT60\SERVER directory) in a text editor. Set the SECURITYTNSNAME parameter using the following syntax:

   SECURITYTNSNAME="sec_rep"

   where sec_rep is the TNS names alias of the WebDB server instance defined in the tnsnames.ora file.

2. Save and close the Reports Server configuration file.

3. Stop and restart the Reports Server to accept the changes made to the Reports Server configuration file.

5.3.2 Step 2. Create user accounts

You will need to create the following users accounts:

- The Reports Developer system administrator to create and maintain access controls for the restricted Reports Servers, report definition files, and printers. See Section 5.3.2.1, “To create the Reports Developer system administrator user account”. Once you create the Reports Developer system administrator, you can start creating access controls for the Reports Servers, report definition files, and printers you want to restrict.

- Any user who will be given access privileges to run a restricted report to a restricted Reports Server and printer. See Section 5.3.2.2, “To create users accounts for running reports”. You can create user accounts at anytime. However, if you are restricting access to the Reports Server, report definition files, or printers, and know which users should have access to them, it is best to create the users accounts first.
5.3.2.1 To create the Reports Developer system administrator user account

In order to perform security administration in WebDB, you must have a user account that is assigned the RW_ADMINISTRATOR role. Only those users with the RW_ADMINISTRATOR role can access Reports Developer Security wizards in WebDB. In addition, you must have BUILD IN privileges to the schema that will own the report’s packaged procedure and any list of values (LOV) that you may create. If you have a user account with DBA privileges, then you can create user accounts. Otherwise, contact your DBA and request that user accounts be created.

1. For WebDB online help on this task, click the help button on the title bar...
2. Click and type... Creating user accounts
3. Click Find and then click... Creating user accounts to display the topic.

Note: It is possible that the packaged procedures and the parameter list of values that you create may be owned by different schemas. You may need BUILD IN privileges to more than one schema.

To add a report item to a WebDB site, a WebDB site must be created. If you will be responsible for creating the WebDB site, you must be a DBA with Execute privileges on the SYS.DBMS_SQL packaged procedure with the Grant option. This privilege will allow you to create the site and grant Manage Item privileges to other users.

Tip: If someone else is the site administrator, you must be given Own, Manage Item, or Create With Approval privileges for the folder that you want to add items to. Contact the DBA or site administrator for more information.

5.3.2.2 To create users accounts for running reports

Any users who will be given access privileges to run report requests must have a user account that WebDB can recognize. Reports Developer has four predefined roles that can be assigned to users. Each role gives users access to certain administrative controls, such as monitoring jobs or viewing error messages. By default, Reports Developer basic user functions (that is, the RW_BASIC_USER role) are implied if users are not assigned specific Reports Developer roles.
If you have a user account with DBA privileges, then you can create user accounts. Otherwise, contact your DBA and request that user accounts be created.

For this exercise, create or request a user account for Jeff Abers, one of the employees who participates in the 401K plan. His user account should be JABERS. He is assigned the basic user role. Contact your DBA to create user accounts for those users who require access privileges to run report requests. Assign users Reports Developer roles as needed.

Tip: If the JABERS user account already exists, then append your initials to it (for example, JABERSAA).

5.3.3 Step 3. Create availability calendars

An availability calendar determines when report definition files, Reports Servers, and printers are available for processing.

You can create two types of availability calendars:

- A simple availability calendar defines a single availability rule (for example, daily, Sunday through Saturday from 12:00 a.m. to 10:00 p.m.).

- A combined availability calendar combines two or more availability calendars (for example, combining the daily calendar with a maintenance calendar) into a single availability calendar.

You can associate only one availability calendar with a report definition file, Reports Server, or printer. If your production environment requires more than one availability rule, you will need to combine availability calendars.

Note: Availability calendars are not necessary if the reports definition files, Reports Servers, and printers are always available for processing.
In this exercise, you will create a production calendar that determines the availability for every day of the week, days with scheduled maintenance, and holidays. To do this, you will create the following availability calendars:

- Daily calendar with an availability period of every Sunday through Saturday from 12:00 a.m. to 10:00 p.m.
- Maintenance calendar with an availability period of every Saturday from 3:00 p.m. to 10:00 p.m.
- Christmas calendar with an availability period starting on December 25 at 12:00 a.m. and ending on December 26 at 12:00 a.m.
- Production calendar that combines all the above calendars, and then excludes the maintenance and Christmas calendars. Excluding these calendars prohibits processing based on their availability rules.

5.3.3.1 To create the daily calendar
You will create a daily calendar with an availability period of Sunday through Saturday from 12:00 a.m. to 10:00 p.m.

Tip: Click to access context-sensitive help for the current wizard page. Click on the title bar to access the WebDB help system.

1. Access WebDB and log on as the Reports Developer system administrator.
   Tip: You must be logged on as the Reports Developer system administrator to access the Reports Developer Security wizards.

2. On the Oracle WebDB home page, click Administer.
   Tip: You also can click from the navigation toolbar on any WebDB page to access the Administer page.

5. On the Availability Calendars page, click the Simple Availability Calendar option to create a new calendar.
6. Click Create to create the simple Availability calendar.
7. On the Simple Availability Calendar page, type daily as the Calendar Name.
   Tip: If the Daily calendar already exists, then append your initials to it (for example, DailyAA).
8. Click .

9. On the Date/Time Availability page, specify today’s date as the start month, date, and year, and 12:00 a.m. as the start time.

10. Specify today’s date as the end month, date, and year, and 10:00 p.m. as the end time.

11. Choose Daily as the Repeat option. This will repeat the duration pattern every day. For example, if the start date is Monday, January 4, 2000, then this pattern will repeat every day starting on this date until the pattern is terminated.

12. Click .

13. (Optional) On the Simple Availability Calendar Summary page, click Show Calendar to view a visual representation of the daily calendar. Green indicates availability. Close the calendar when you are finished reviewing it.

14. Click .

15. On the Create Simple Availability calendar page, click OK to create the calendar.

5.3.3.2 To create the maintenance calendar

You will create a maintenance calendar with an availability period of every Saturday from 3:00 p.m. to 10:00 p.m. In a later step, you will add this calendar to the Production calendar and then exclude it to prohibit processing based on the date and time specified.

1. From the Availability Calendars page, click the Simple Availability Calendar option to create a calendar.

2. Click Create.

3. On the Simple Availability Calendar page, type Maintenance as the Calendar Name.

   Tip: If the Maintenance calendar already exists, then append your initials to it (for example, MaintenanceAA).

4. Click .
5. Follow steps 9-15 in Section 5.3.3.1, "To create the daily calendar" to define the following availability rule:

**Table 5–2 Maintenance calendar rule**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration Start</td>
<td>Specify a date starting on a Saturday (for example, January 8, 2000), and time starting at 3:00 p.m.</td>
</tr>
<tr>
<td>Duration End</td>
<td>Specify the same date defined as the start date, and time ending at 10:00 p.m.</td>
</tr>
<tr>
<td>Repeat</td>
<td>Choose <strong>Weekly</strong>.</td>
</tr>
</tbody>
</table>

**5.3.3.3 To create the Christmas calendar**

You will create a Christmas calendar with an availability period of every December 25 from 12:00 a.m. to December 26 at 12:00 a.m. In a later step, you will add this calendar to the Production calendar and then exclude it to prohibit processing based on the date and time specified.

1. From the Availability Calendar page, click the **Simple Availability Calendar** option to create the third calendar.

2. Click **Create**.

3. On the Simple Availability Calendar page, type **Christmas** as the Calendar Name.

   **Tip:** If the Christmas calendar already exists, then append your initials to it (for example, ChristmasAA).

4. Click ** ». **

5. Follow steps 9-15 in Section 5.3.3.1, "To create the daily calendar" to define the following rule:

**Table 5–3 Christmas calendar rule**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration Start</td>
<td>Specify December 25 and 12:00 a.m.</td>
</tr>
<tr>
<td>Duration End</td>
<td>Specify December 26 and 12:00 a.m.</td>
</tr>
<tr>
<td>Repeat</td>
<td>Choose <strong>Yearly</strong>.</td>
</tr>
</tbody>
</table>
5.3.3.4 To create a combined availability calendar

In this exercise, you will create a Production calendar that combines the Daily, Maintenance, and Christmas calendars, then excludes the Maintenance and Christmas calendars, which prohibits processing based on their availability rules.

1. From the Availability Calendar page, click the Combined Availability Calendar option to create the calendar that will combine the three calendars you created into one.

2. Click Create.

3. On the Combined Availability Calendar page, type Production as the Calendar Name.
   
   **Tip:** If the Production calendar already exists, then append your initials to it (for example, ProductionAA).

4. Click .

5. On the Select Availability Calendars page, ctrl-click the Daily, Maintenance, and Christmas calendars from the Availability Calendars list box.

6. Click to move the selected calendars to the Selected Availability Calendars list box, or click to select all available calendars.

7. Click .

8. On the Exclude Availability Calendars, ctrl-click the Maintenance and Christmas calendars in the Availability Calendars list box.

9. Click to move the Maintenance and Christmas calendars to the Excluded Availability Calendars list box. Doing so prohibits processing on the date and time specified in each calendar.

10. Click .

11. On the Combined Availability Calendar Summary page, click Show Calendar to view a visual representation of the availability calendar. Green indicates availability. Close the calendar when you are finished reviewing it.

   **Tip:** It is a good practice to check the combined calendar at this point. You can verify that the calendars you prohibited processing on are excluded during the period specified. Scroll to December to ensure that December 25 is excluded from processing. Choose the Day option and scroll to a Saturday to ensure that processing is unavailable from 3 p.m.
12. Click 

13. On the Create Combined Availability Calendar page, click OK to create the Production calendar in WebDB.

5.3.4 Step 4. Add access to a Reports Developer printer in WebDB

Printer Access defines the following:

- What printer is available in WebDB to print report output.
- Who has access privileges to print output to this printer.
- When this printer is available to print report requests.

Prerequisite: You must already have a printer that Reports Developer can recognize installed and running. Refer to the DESNAME and DESFORMAT command line arguments described in Appendix A, "RWCLI60 Command Line Arguments" for more information.

Tip: Click ? to access context-sensitive help for the current wizard page. Click on the title bar to access the WebDB help system.

1. Click Reports Developer Security from the link history, which is located just above the Navigation toolbar.


3. Click Create to add printer access to WebDB.

4. On the Printer Name page, type Reports_Printer in the Printer Name field.

   Tip: If this printer name already exists, then append your initials to it (for example, Reports_PrinterAA).

5. Type the operating system name of the printer in the OS Printer Name field (for example, the OS printer name in Windows NT might be \net_machine\my_printer). Refer to your operating system’s documentation for more information.

6. Click 

7. On the Users and Roles page, choose JABERS and your Reports Developer system administrator user account from the All Users list box to specify who can output reports to this printer.

8. Click to move this user to the Selected Users list box.

9. Click .
10. On the Availability Calendar page, type `Production` as the availability calendar, or click ![calendar icon] to find the availability calendar.

   **Tip:** If you want to make this printer available all the time, do not specify a calendar.

11. Click ![add printer icon].

12. On the Add Printer Access page, click **OK** to add access to this printer in WebDB.

### 5.3.5 Step 5. Add access to the Reports Server in WebDB

Reports Server Access defines the following in WebDB:

- What Reports Server is available in WebDB for processing requests.
- What printer is available to the Reports Server.
- Who has access privileges to send report requests to this Reports Server.
- When this Reports Server is available to accept report requests.

**Tip:** Click ![help icon] to access context-sensitive help for the current wizard page. Click on the title bar to access the WebDB help system.

1. Click **Reports Developer Security** from the link history.
2. At the Reports Developer Security page, click **Server Access**.
3. Click **Create**.
4. On the Server Name and Printers page, type `repserver` in the Server Name field.

   **Tip:** If this server name already exists, then append your initials to it (for example, RepserverAA).

5. Type `repserver` in the Reports Server TNS Name field.

   **Tip:** The Reports Developer TNS name is the Reports Server service entry name that is added to the `tnsname.ora` file when you installed and configured the Reports Server. See Chapter 3, "Configuring the Reports Server" for more information.
6. Type the Reports Developer Web Gateway URL in lowercase:

   http://my_webserver/cgi-bin/rwcgi60.exe

   **Tip:** The Reports Developer Web Gateway URL is determined by the virtual location of the Web CGI. See Section 3.2.2, "Step 2. Configuring the Web server".

7. Choose the `Reports_Printer` from the Printers list box.

8. Click ✅.

9. On the Users and Roles page, choose JABERS and your Reports Developer system administrator user account from the All Users list box to specify who can access this server.

   **Tip:** Be sure that you select the same users who have been given access to the printer.

10. Click ✅ to move this user to the Selected Users list box.

11. Click ✅.

12. On the Availability Calendar page, type *Production* as the availability calendar, or click ✅ to find the availability calendar.

   **Tip:** If you want to make this server available all the time, do not specify a calendar.

13. Click ✅.

14. On the Add Server Access page, click **OK** to add server access to WebDB.

### 5.3.6 Step 6. Add access to the report definition file in WebDB

Report Definition File Access defines the following in WebDB:

- What Reports Developer .RDF, .REP, or .XML file you want to make accessible in WebDB.
- Who has access privileges to run this report definition file.
- When this report definition file is available to run.
- How report output is delivered by restricting the report request options (that is, required and option parameters) that are available to users at runtime. This includes specifying the Reports Servers and printers that are available to users.
In this exercise, you will restrict access to the `security.rdf` file (located in the `ORACLE_HOME\TOOLS\DOC60\US\RBR60` directory) in WebDB based on the following information:

- Destination type is restricted to Cache and Printer.
- Destination format is restricted to HTMLCSS and PDF.
- P_LASTNAME user parameter is restricted to a list of values defined in WebDB.
- P_SSN user parameter is used to validate the social security number and last name pair.
- COPIES system parameter is restricted to two copies preventing users from printing more than two copies of their report.
- USERID system parameter is restricted so that users can save database logon information in the Runtime Parameter Form. Specifying the USERID as a restricted parameter is necessary if you want users to schedule their reports to run automatically.

**Tip:** Click ![context-sensitive help icon](image) to access context-sensitive help for the current wizard page. Click ![WebDB help system icon](image) on the title bar to access the WebDB help system.

### 5.3.6.1 To create a list of values for the lastname user parameter

If you want users to select values from a list of values for any system or user parameters you define on the Optional Parameters page, you must create this list in WebDB.

Recall that the `security.rdf` report gathers information about the vested portfolios of employees participating in the company’s 401K plan. You want to restrict access to only those employees who participate in the plan. In this exercise, you will create a list of values for the P_Lastname user parameter that lists the last names of these employees.
Tip: If you are not publishing the report request on a WebDB site, then creating a list of values in WebDB is not necessary. You can create a list of values in Report Builder using the Parameter Form editor:

1. For Report Builder online help on this task, choose Help → Report Builder Help Topics
2. On the Index page, type... parameter, list of values
3. Then click Display to view help topic... Creating a list of values (LOV) for a parameter

1. Click from the navigation toolbar.
2. At the Shared Components menu, click Lists of Values (LOV).
3. Choose the Static - Static Values option, and then click Create LOV.
4. On the Create Static List of Values page, choose a schema as the Owning Schema of this LOV.
5. Choose PUBLIC from the Privileges list box so that all users have access to this LOV.
6. Type LASTNAME_LOV as the name of LOV.
   Tip: If this LOV already exists, then append your initials to it (for example, LASTNAME_LOVAA).
7. Choose Combo Box as the Default Format.
8. Enter the following values in the table:

   Table 5–4 Static List of Values
<table>
<thead>
<tr>
<th>Display Value</th>
<th>Return Value</th>
<th>Display Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abers</td>
<td>Abers</td>
<td>1</td>
</tr>
<tr>
<td>Costner</td>
<td>Costner</td>
<td>2</td>
</tr>
<tr>
<td>Matsumoko</td>
<td>Matsumoko</td>
<td>3</td>
</tr>
<tr>
<td>Williams</td>
<td>Williams</td>
<td>4</td>
</tr>
</tbody>
</table>
9. Click Add LOV.
10. From the Manage List of Values page, the newly created LOV is displayed in the Recently Edited List of Values section. If you want to test the LOV, you can do so here.

5.3.6.2 To add access the report definition file

Tip: If you back out of the wizard page (that is, click Back on your Web browser), you will lose the settings you defined on that page. If you need to make changes, first create the packaged procedure for the report by completing the wizard. Then, edit the package by clicking Edit on the Manage Component page.

1. Click from the navigation toolbar and click Reports Developer Security.
3. Click Create.
4. On the Report Name and Schema page, choose a schema from the Owner list box. The schema that you choose will own this packaged procedure for this report.
5. Type Investment_Report in the Report Name field.
   Tip: The report name cannot be prefaced with numeric characters (for example, 401K_report is an invalid file name and my_401K_report is valid).
   Tip: If this report name already exists, then append your initials to it (for example, Investment_ReportAA).
6. Choose repserver from the Reports Servers list box.
7. Type security.rdf as the Reports Developer File Name.
   Tip: Ensure that the Reports Server can find this report definition file. The report’s source path must be set in the SOURCEDIR parameter in the Reports Server configuration, or must be set in the REPORTS60_PATH environment variable.
8. Click .
9. On the Users and Roles page, choose JABERS and your Reports Developer system administrator user account from the All Users list box to specify who can run this report.

   **Tip:** Be sure that you choose the same users who have been given access to the printer and Reports Server.

10. Click \( \text{ } \) to move this user to the Selected Users list box.

11. Click \( \text{ } \).

12. On the Availability Calendar page, type `Production` as the availability calendar, or click \( \text{ } \) to find the availability calendar.

   **Tip:** If you want to make this report definition file available all the time, do not specify a calendar.

13. Click \( \text{ } \).

14. On the Required Parameters page, ctrl-click Cache and Printer from the Types list box.

15. Shift-click HTMLCSS and PDF from the Formats list box.

16. Choose `Reports_Printer` from the Printers list box.

   **Tip:** If the printer you defined does not appear, you may have entered an incorrect OS Printer Name when you created access to your printer. Finish creating this report definition file package. It is likely that an invalid package will be created. Return to the Printer Access wizard and edit access to the `Reports_Printer`. After you edit the printer access, return to the Report Definition File Access wizard, edit the report definition file access for this report, then create a new package.

17. (Optional) Choose another Parameter form template. The template you choose will determine the page style on which the Runtime Parameter Form is displayed.

18. Click \( \text{ } \).

19. On the Optional Parameters page, type `P_LASTNAME` in the Parameter Name column. When users run this report at runtime, they will be required to select a last name to run the report.

   **Tip:** `P_LASTNAME` is the name of the parameter defined in report. Open the `security.rdf` file in Report Builder and view the parameters in the Parameter Form editor to determine the parameter's name.
20. Type `LASTNAME_LOV` in the LOV column to enable users to choose the last name of the 401K participant from a list of values, or click to find the LOV.

21. Type `P_SSN` in the second row of the Parameter Name column to require users to type their social security number in the Runtime Parameter Form.

22. Type `COPIES` in the third row of the Parameter Name column to restrict the number of copies the user can print when outputting the report to a printer.

23. Type `1` in the Low Value column.

24. Type `2` in the High Value column.

25. Type `USERID` in the fourth row of the Parameter Name column. This will enable users to specify the database that they can connect to if they want to schedule the report to run automatically.

26. Click twice to skip the Validation Trigger page.

27. At the Add Report Definition File Access page, click OK to create the packaged procedure for this report. When the package is created, the Manage Component page appears. From this page, you can edit the report access, run the report, or set up the Parameter Form. The next exercise explains how to set the default parameter values in the Parameter Form that are used to run the report.

   Tip: If an invalid package is created, you will be unable to proceed to the next step. Verify the access controls that you defined for the printer, Reports Server, and report. Make the necessary changes and then try to create a valid production package for this report definition file.

   To edit access to the report definition file, click from the navigation toolbar. At the Reports Developer Security menu, choose Report Definition File Access. Then, to access the Manage Component page for a particular report, find the report or choose the report from the Recently Edited Report Definition File section. At the Manage Component page, click Edit.

5.3.7 Step 7. Set parameter values on the Reports Developer parameter form

As the Reports Developer system administrator, you can run the restricted report request you just created to ensure that it will run as expected. You also can set the default parameters that will be available to users at runtime. You can run and set default parameter values from the Manage Component page.
5.3.7.1 To run the report output to cache

In this exercise and the next, you will choose parameters values to run the report to cache for debugging purposes, not to set the default values that will be available to users at runtime. You will set the default values in Section 5.3.7.3, "To set the default parameters for users at runtime".

1. On the Manage Component page, click Parameters to set the default parameters and choose the parameters that will be visible on the Runtime Parameter Form.

   **Tip:** To access the Manage Component page, click . At the Reports Developer Security menu, choose Report Definition File Access. Then, find the report or choose the report from the Recently Edited Report Definition File section.

2. On the Reports Developer Parameter Form, set the following parameters:

   **Table 5–5 Parameter form settings for debugging cache output**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>repserver</td>
</tr>
<tr>
<td>Printer</td>
<td>Reports_Printer</td>
</tr>
<tr>
<td>Destype</td>
<td>Cache</td>
</tr>
<tr>
<td>Desformat</td>
<td>HTMLCSS</td>
</tr>
<tr>
<td>Desname</td>
<td>blank</td>
</tr>
<tr>
<td>Copies</td>
<td>1</td>
</tr>
<tr>
<td>P_LASTNAME</td>
<td>Aber</td>
</tr>
<tr>
<td>P_SSN</td>
<td>559014203</td>
</tr>
<tr>
<td>USERID</td>
<td>username/password@my_db</td>
</tr>
</tbody>
</table>

   where username/password@my_db is the username and password for the database you want to connect to.

   **CAUTION:** When setting parameter values for debugging purposes, be sure to delete (or not save) any confidential parameter values, such as social security numbers, from this Parameter Form. Otherwise, this confidential information will be made public when you add this report request to a WebDB site.

3. Click Run Report to run the report as requested.
5.3.7.2 (optional) To run the report output to a restricted printer

1. If you want to send the output to the printer, return to the Manage Component page and click Parameters.

2. At the Reports Developer Parameter Form, choose the following parameter values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>repserver</td>
</tr>
<tr>
<td>Printer</td>
<td>Reports_Printer</td>
</tr>
<tr>
<td>Destype</td>
<td>Printer</td>
</tr>
<tr>
<td>Desformat</td>
<td>PDF</td>
</tr>
<tr>
<td>Desname</td>
<td>defaults to the printer name</td>
</tr>
<tr>
<td>Copies</td>
<td>1</td>
</tr>
<tr>
<td>P_LASTNAME</td>
<td>Abers</td>
</tr>
<tr>
<td>P_SSN</td>
<td>559014203</td>
</tr>
<tr>
<td>USERID</td>
<td>username/password@my_db</td>
</tr>
<tr>
<td></td>
<td>where username/password@my_db is the username and password for the database you want to connect to.</td>
</tr>
</tbody>
</table>

3. Click Run Report.

4. Click OK when a message appears stating that the report was printed successfully.
5.3.7.3 To set the default parameters for users at runtime

Once you are satisfied that the report can run based on the restrictions imposed, you can set the default parameter values and choose the parameters that will be available to users on the Runtime Parameter Form.

1. On the Reports Developer Parameter Form, set the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Visible to User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>repserver</td>
<td>No</td>
</tr>
<tr>
<td>Printer</td>
<td>Reports_Printer</td>
<td>No</td>
</tr>
<tr>
<td>Destype</td>
<td>Cache</td>
<td>Yes</td>
</tr>
<tr>
<td>Desformat</td>
<td>HTMLCSS</td>
<td>Yes</td>
</tr>
<tr>
<td>Desname</td>
<td>blank</td>
<td>No</td>
</tr>
<tr>
<td>COPIES</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>P_LASTNAME</td>
<td>blank</td>
<td>Yes</td>
</tr>
<tr>
<td>P_SSN</td>
<td>Type your SSN</td>
<td>Yes</td>
</tr>
<tr>
<td>USERID</td>
<td>Type the database logon</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Tip: You may want to make parameters visible to users on the Runtime Parameter Form only when they need to take an action on the parameter (that is, select or input a value) to run the request. In this case, the Server and Printer parameters are restricted to one server and printer. The Desname parameter is populated automatically with the printer name when Printer is chosen as the Destination type. These parameters do not require user input to run the report.

When users run the report from a WebDB site, they can set the default parameters values available to them on the Runtime Parameter Form to their personal preferences. See Section 5.3.8, “Step 8. Make the report available to users” for more information.

2. Click Save Parameters to save the changes made to this Parameter Form.
5.3.8 Step 8. Make the report available to users

You make the report available to users in a WebDB site by adding a link as a WebDB component that points to the INVESTMENT_REPORT packaged procedure.

5.3.8.1 To create a WebDB site

1. Create a WebDB site if it has not already been created

   1. For WebDB online help on this task, click the help button on the title bar...
   2. Click and type... web site
   3. Click Find and click... Creating web sites to display the topic.

   Tip: To create a WebDB site, the Reports Developer system administrator will need to have site administrator privileges (that is, a DBA with execute privileges on the SYS.DBMS_SQL packaged procedure with the Grant option). If someone else is the site administrator, ask that person to create the WebDB site.

5.3.8.2 To create a folder in the WebDB site

You will create the folder in which the report’s packaged procedure is added. By default this folder and any items that are added to it are available only to the owner of the folder (that is, the Reports Developer system administrator). You can make the folder available to all users (that is, to public users) or available only to users who have been given access to it. You will restrict access to this folder only to the users who have access privileges to run this report (that is, JABERS).

   Tip: If you make a folder public, PUBLIC users (that is, users who have not logged on to the WebDB site) can access the report’s Parameter/Scheduling form and may unknowingly save their personal information to it. Subsequent PUBLIC users will see this confidential information. To prevent this from happening, it is best to restrict access to the folder to those users who have access to run the report. Users must log on to access the restricted folder. Once logged on, the information they save on the Parameters/Scheduling form is secured (that is, only they can view it).

   Tip: Click to access context-sensitive help for the current wizard page. Click on the title bar to access the WebDB help system.
1. From your Web browser, type the URL of the WebDB site. For example:

   http://my_webdb_server.com:1111/my_webdb_site

   **Tip:** If you have site administrator privileges to create a site, click [Site Home Page](#) from the navigation toolbar on any WebDB page to access the Sites page. Click Site Home Page to access the WebDB site.

2. Log on as the Reports Developer system administrator.

   **Tip:** To add a WebDB component to a WebDB site, your Reports Developer system administrator user account must have site administrator privileges (that is, a DBA with execute privileges on the SYS.DBMS_SQL packaged procedure with the Grant option). If you do not have site administrator privileges, you must have Own, Manage Item, or Create with Approval privileges for the folders in which the component is being added. Contact your DBA or site administrator for more information.

3. At the WebDB Site home page, click [Edit](#).

4. Click [Add Folder](#) to add a new folder to your site.

5. On the Folder Manager page, type **Benefits** as the internal folder name of the new folder.

6. Type **Benefits** as the title of the folder that will be displayed in the WebDB site.

   **Tip:** If the Benefits folder has already been created by another user, then append your initials to the folder name (for example, BenefitsAA).

7. Click [Create](#) to create the folder.

8. Choose Benefits from the list box.

9. Click [Edit](#).

10. Click the [Users](#) tab.

11. On the Benefits page, type **JABERS** as the Name of the user you want to have access to this folder.

12. Click [Add to Access list](#). Notice that JABER is listed in the User Access List with view privileges. Keep this default.

13. Click [Finish](#) to return to the Benefits folder.
5.3.8.3 To add the report request to the folder

1. On the Benefits, click to access the Item wizard and add the report request to this folder.
2. On the Add an Item page, choose WebDB Component as the Item Type.
3. Choose Regular Item as the Display Option.
4. Click Next.
5. On the WebDB Component page, choose WEBDB.INVESTMENT_REPORT from the list box, where WEBDB is the name of the schema that owns this report’s package procedure for the 401K report.
6. Type Investment Summary Report as the Title.
7. Choose General as the Category.
8. Type Restricted 401K Report in the Description text box.
9. Click Next.
10. On the second WebDB Component page, choose the Display Parameter Form option.
11. Click Finish. A link to the packaged procedure that contains the report request appears in the Benefits folder.

5.3.8.4 To run this report as a user

You will run this report as JABERS, not as the Reports Developer system administrator. In this exercise, you will set your default parameter settings for Jeff Abers and then run the report.

Tip: Click to access context-sensitive help for the current wizard page. Click on the title bar to access the WebDB help system.

1. Click Log Off on the navigation bar to log off as the Reports Developer system administrator.
2. Click Log On to log on as JABERS.
3. Click Site Map to access the Benefits folder.
4. Click Benefits folder.
5. In the Benefits folder, click **Investment Summary Report**.

6. On the Parameters/Scheduling page, choose the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destype</td>
<td>Cache</td>
</tr>
<tr>
<td>Desformat</td>
<td>HTMLCSS</td>
</tr>
<tr>
<td>COPIES</td>
<td>1</td>
</tr>
<tr>
<td>P_LASTNAME</td>
<td>Abers</td>
</tr>
<tr>
<td>P_SSN</td>
<td>559014203</td>
</tr>
<tr>
<td>USERID</td>
<td>username/password@my_db</td>
</tr>
</tbody>
</table>

   where `username/password@my_db` is the username and password for the database you want to connect to.

7. Click **Save Parameters** to save your personalized settings.

   Note that the default settings saved here are the ones that are accessible only to this user. If you (or someone else) logged on as a different user, the default settings defined by the Reports Developer system administrator would display. That user could then personalize her or his own settings.

8. Click Run Report.

### 5.3.9 Step 9. Schedule the report to run and push the output to a WebDB site

Suppose that Jeff Abers only wants to review his 401K investments once a month. Further, he prefers to have this report run automatically and pushed to his own personal folder by 9:00 a.m. on the last Friday of every month. First, you will create Jeff’s own personal folder (that is, one that only his user account can access). Then, you will schedule the report to run automatically.

**Prerequisite:** You must have already completed the exercise in Section 5.3.8, “Step 8. Make the report available to users”.

---

Table 5–8  **User’s default parameter settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destype</td>
<td>Cache</td>
</tr>
<tr>
<td>Desformat</td>
<td>HTMLCSS</td>
</tr>
<tr>
<td>COPIES</td>
<td>1</td>
</tr>
<tr>
<td>P_LASTNAME</td>
<td>Abers</td>
</tr>
<tr>
<td>P_SSN</td>
<td>559014203</td>
</tr>
<tr>
<td>USERID</td>
<td>username/password@my_db</td>
</tr>
</tbody>
</table>

where `username/password@my_db` is the username and password for the database you want to connect to.
5.3.9.1 To create a personal folder

To ensure that only the specified user can access his or her own personal reports, the user (that is, you are logged on as JABERS for this exercise) can create her or his own personal folder.

**Tip:** Click ? to access context-sensitive help for the current wizard page. Click on the title bar to access the WebDB help system.

1. Access the WebDB site from your Web browser, and log on as JABERS if you have not already done so.
2. Click **Administration** from the navigation bar.
3. Under the Access Managers section, click **Personal Information**.
4. Check the Create Personal Folder box. If the Create Personal Folder box does not appear, then your personal folder has already been created.
5. Type personal information as desired.
6. Click .

You are the owner of your personal folder. No one else can access it unless you give them permission to do so. You are ready to schedule the 401K report to run automatically and push to the JABERS personal folder.

5.3.9.2 To schedule the report

In this exercise, you will schedule the report to run every last Friday of the month at 9:00 a.m. You also want to retain historical records of your 401K results for two months.

**Tip:** Click ? to access context-sensitive help for the current wizard page. Click on the title bar to access the WebDB help system.

1. Click **Site Map** from the navigation bar.
   **Tip:** You should be logged on as JABERS.
2. Click the Benefits folder.
3. Click **Investment Summary Report**. If the default parameters have not been set, then go to the Section 5.3.8.4, "To run this report as a user" and set the default parameters.
4. On the Parameters/Scheduling page, click **Schedule**.
5. Choose to start the job at 9:00 a.m. on today’s date.
   **Tip:** If you want to run the report immediately, choose the Immediately option.

6. Choose the following Repeat option:
   Last **Friday** of each month on or before the 30th.
   **Tip:** Rather than waiting until the end of the month for the report to run, set the repeat option to repeat every n hours. Once you are satisfied that the report output can be pushed successfully to the result folder, reset the repeat pattern.

7. Set the following output destination options:

<p>| <strong>Table 5–9 Output destination settings</strong> |</p>
<table>
<thead>
<tr>
<th><strong>Field</strong></th>
<th><strong>value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Site</td>
<td>The name of the WebDB site in which the Result folder is located.</td>
</tr>
<tr>
<td>Log File Folder</td>
<td>JABERS</td>
</tr>
<tr>
<td>Result Title</td>
<td>My 401K Report</td>
</tr>
<tr>
<td>Result Folder</td>
<td>JABERS</td>
</tr>
<tr>
<td>Overwrite Previous Result</td>
<td>uncheck</td>
</tr>
<tr>
<td>Expiration</td>
<td>60 days</td>
</tr>
</tbody>
</table>

**CAUTION:** The names of the Log File Folder and Result Folder are case-sensitive. If want your report output and status information to be pushed to an existing folder, you must type the exact folder name. If you mistype the folder names, WebDB will not be able to find them, and by default will add the named folders to the Reports Developer Output and Reports Developer Status folders. By default, these folders are given public access (that is, all users will be able to view your personal report). Exercise care when defining these folders.

8. Click **Submit**. A message stating that the report was successfully scheduled appears.
   **Tip:** If you scheduled the report to run immediately, the report output is displayed in your browser.

9. Click **OK**. The job will run based on its scheduled date, time, and repeat pattern.
5.3.9.3 To view the pushed report output

1. Click **Site Map** from the navigation bar.
   **Tip:** You are still logged on as JABERS.

2. Click **JABERS** to open the folder.

3. Click **My 401K Report** to view the report.

   Notice that in addition to a link to the report itself, a link to status information about the report is also available. Use this status link to help you troubleshoot any problems you may have running this scheduled report. Depending on the Reports Developer role (for example, RW_BASIC_USER) this user is assigned, you may see different status details. If users are having problems scheduling and running their reports, they should contact the Reports Developer system administrator for help.

Optional exercise

Suppose that the Human Resource director asked you (the Reports Developer system administrator) to make a stock report available to all employees. You want to run this stock report automatically every morning so that employees can monitor the status of certain stocks. This report will be pushed to a public folder from which all employees can view it.

Use the exercises in this chapter to help you add access to the `template.rdf` report (located in the `ORACLE_HOME\TOOLS\DOC60\US\RBR60` directory) in WebDB.

**Tip:** Since you (as the Reports Developer system administrator) will be scheduling this report to run and push to a public folder, this report needs to be accessible only to the Reports Developer system administrator.

Add this report’s packaged procedure to the Benefits folder as a WebDB component. Then, schedule this report to run every morning at 10:00 a.m. pushing the report output to a new folder called Stocks.

**Tip:** The Stock folder must be set up to display for public users.
5.3.10 Summary

You have successfully configured the Reports Server for access control. In this chapter, you learned how to do the following:

- Create availability calendars to determine when reports, Reports Servers, and printers are available for processing.
- Add access to printers, Reports Servers, and report definition files in WebDB by restricting who can access them and restricting when they are available for processing.
- Add a report request, stored as a packaged procedure, to a WebDB site as a WebDB component.
- Enable users to set their personal default parameters and run report requests from a WebDB site.
- Enable users to schedule reports to run automatically and push the resulting report to their own personal folder.
Configuring Reports Server Clusters

This chapter will show you how to configure the Reports Server in a cluster to improve performance and loading balancing. This becomes important as the need to deliver information to a rapidly growing user base becomes more demanding.

Reports Server clustering addresses this demand by leveraging your organization’s existing hardware investment by plugging in additional application servers as they are needed. This enables the processing capabilities of your Reports Servers to grow as your organization grows.

Before you begin to configure your Reports Server for clustering, you should be familiar with the basic Reports Server architecture. See Chapter 2, "Publishing Architecture and Concepts" for more information. You must also have already set up your Reports Server using a basic configuration. See Chapter 3, "Configuring the Reports Server" for more information.

6.1 Clustering Overview

Suppose that you have three machines configured as Reports Servers that you want to cluster. These machines are described below:

<table>
<thead>
<tr>
<th>Machine/Server TNS name</th>
<th>Description</th>
<th>Master/Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT-1</td>
<td>4 CPU NT server</td>
<td>Master</td>
</tr>
<tr>
<td>NT-2</td>
<td>2 CPU NT server</td>
<td>Slave</td>
</tr>
<tr>
<td>SUN-1</td>
<td>2 CPU Sun Solaris workstation</td>
<td>Slave</td>
</tr>
</tbody>
</table>

Note: The decision to make the NT-1 machine the master server was arbitrary. The number of CPUs was not a determining factor.
For step-by-step instructions on configuring the Reports Servers in a cluster as described in this overview, see Section 6.2, "Configuring Reports Servers in a Cluster Example".

You will designate NT-1 as the master, then set the CLUSTERCONFIG parameter to enable this server to recognize NT-2 and SUN-1 as slaves. To simplify this example, the MAXENGINE and MINENGINE for the master and each slave server are set to the number of CPUs available on each machine.

Once configured, you will send report requests to the master server (that is, SERVER=NT-1) which redirects the reports to the slaves. When the master server is started, it checks the configuration file. The master contacts each of the slave servers in the order that they are listed in the configuration file and notifies them to start up the defined number of engines (for example, two engines each). When the slave engines are started, they are under the control of the master, which allocates jobs to them using a round-robin algorithm.

Suppose that the master server (that is, NT-1) receives seven report requests. The master uses its four engines to run the first four reports. For the fifth and sixth reports, the master redirects the requests to the two NT-2 engines to run them. When the master receives the seventh report, it redirects the request to the first SUN-1 engine to run it. All output is written to a central cache (that is, one that is shared by all servers). The master sends the output back to the requestor (for example, a Web browser).

Note: It is possible for slave servers to remain fully functional Reports Servers in their own right if they can start engines independently of the master server. Suppose that the MAXENGINE and MINENGINE parameters of the NT-2 Reports Server configuration are set to three. This means that three engines are dedicated to the NT-2 Reports Server and can receive requests without the master’s knowledge. When configured as a slave server (that is, the MAXENGINE and MINENGINE parameters in the master configuration for NT-2 are set to two), the NT-2 Reports Server has a total of five engines started: three engines that are dedicated to the NT-2 server and two engines are dedicated slaves to the master.
6.2 Configuring Reports Servers in a Cluster Example

This section provides step-by-step instructions for configuring Reports Servers clusters. This example describes the following:

■ enabling communication between the master and slaves
■ configuring the master server
■ running report requests to clustered servers
■ resubmitting jobs when an engine goes down
■ adding a server to an existing configuration

In this example, you will configure the server machines for clustering as described in Table 6–1, "Example server machines descriptions".

The following assumptions have also been made for each machine:

■ The Oracle Reports Server component has been installed.
■ The Reports Server has been configured using the machine name as the TNS service entry name (for example, NT-1) in the tnsames.ora file. See Section 3.2, "Configuring the Reports Server: Reports Web CGI Example" for more information about the Reports Server configuration.
■ A central file server is running and set up with two directories: a Source directory (where report definition files are stored) and a Cache directory (where all cached report output is sent).

All engines must write their output to a central cache and all engines read report definition files from a central source directory. A central source directory guarantees that all engines are running the same reports. This also eliminates copying updated report definition files to various locations. A central cache enables the master server to serve duplicate jobs and jobs run within the specified tolerance without going to each slave server’s local disk.

■ All engines see the same aliases for printers (unless the output is always being sent to the default printer).
6.2.1 Enabling communication between master and slaves

On the NT-1 machine (master):

1. Open the tnsnames.ora located in the ORACLE_HOME\NET80 directory in a text editor, and add the following:

   nt-2.world=(ADDRESS=(PROTOCOL=tcp)(HOST=nt-2)(PORT=1949))
   sun-1.world=(ADDRESS=(PROTOCOL=tcp)(HOST=sun-1)(PORT=1949))

   where nt-2.world and sun-1.world are the names of the server instances and .world is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.

On the NT-2 machine (slave):

1. Open the tnsnames.ora located in the ORACLE_HOME\NET80 directory in a text editor, and add the following:

   nt-1.world=(ADDRESS=(PROTOCOL=tcp)(HOST=nt-1)(PORT=1949))

   where nt-1.world is the name of the server instance and .world is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.

2. Open the nt-2.ora (the Reports Server configuration file) located in the ORACLE_HOME\REPORT60\SERVER directory, and set the INITEGINE parameter to 0. This ensures that the only engines created at startup are the ones started by the master.

3. Repeat steps 1 and 2 on the SUN-1 server machine. In step 2, edit the sun-1.ora configuration file.
6.2.2 Configuring the master server

In this section you will configure the master using the following settings:

- Edit the master server configuration file to identify the slave servers to the master and to control the number of engines associated with master server.
- Set the parameters in the master server configuration file that defines the following:
  - The engine settings and identifies the cache and source directories.
  - Since there are four CPUs on this machine, you will use four local engines to start at the same time as the server.
  - These four engines will shut down if they are idle for 60 minutes, and will restart after running 50 jobs.
  - The number of processes that can communicate with the server at one time is set to the maximum number of 4096.
- Set the CLUSTERCONFIG parameter to identify the slave servers to the master. In this example, you will start two engines on each slave server when the master is started.

Note that the ENGLIFE and MAXIDLE parameters for the master server’s engines are implied for all slave engines.

On the NT-1 server machine (master):

1. Open `nt-1.ora` (the Reports Server configuration file) located on `ORACLE_HOME\REPORT60\SERVER` directory.
2. Edit the configuration file according to settings below:
   
   ```
   maxconnect=4096
   sourcedir="X:\Source"
   cachedir="X:\Cache"
   cachesize=50
   minengine=0
   maxengine=4
   initengine=4
   maxidle=60
   englife=50
   ```

   **Note:** The NT-1 machine is mapped to the central server on the X: drive.
3. Edit the configuration file according to the settings below:

```
clusterconfig="(server=nt-2
    minengine=0
    maxengine=2
    initengine=2
    cachedir="W:\Cache")
  (server=sun-1
    minengine=0
    maxengine=2
    initengine=2
    cachedir="/share/Cache")"
```

where:

- `server` is the TNS service entry name of the slave server.
- `minengine` is the minimum number of runtime engines this master server should have available to run reports.
- `maxengine` is the maximum number of runtime engines this master server has available to run reports.
- `initengine` is the initial number of runtime engines started by this master server.
- `cachedir` is the central cache directory for this master server.

**Notes**

- Each slave definition must be surrounded by parentheses.
- The cache directory setting for the NT and the UNIX machines are different. Not all servers need to see the shared file system by the same definition (that is, the master is mapped to the X: drive, while the slave is mapped to W: drive).
- The slave servers must have their REPORTS60_PATH environment variable set to `/share/Source` (for the SUN-1 server machine) and set to `W:\Source` (for the NT-2 machine).

4. Shut down and restart the master server so that the master server can recognize the new configuration.

This completes the configuration. Eight engines will start when the master server is started.
6.2.3 Running reports in a clustered configuration

To run report requests to Reports Servers that have been configured for clustering, you specify the master server in the SERVER command line argument (that is, SERVER=NT-1) along with any other relevant arguments for the thin client executable. The master server assigns incoming jobs to the engines on the slave servers.

**Tip:** If you set the REPORTS60_REPORTS_SERVER environment variable to the master server, you can omit the SERVER command line argument. See Appendix C, "Environment Variables" for more information.

See Chapter 4, "Running Report Requests" for more information on the various report request methods you can use.

See Section 6.2.4, "Resubmitting jobs when an engine goes down" if you have problems submitting report requests to the server cluster.

The master server’s jobs can be monitored by using the Queue Viewer in the Queue Manager. Refer to the Queue Manager online help for more information.

6.2.4 Resubmitting jobs when an engine goes down

If an engine goes down while a report is running, the Retry settings defined in the SCHEDULE command line argument dictates whether the job will be re-run or not. If no Retry settings have been specified, then the job is lost. This job failure, however, will be logged against the server log file, and displayed in the list of jobs in the Queue Manager. If the command line includes retry settings, then the master server will re-run the job with the next available engine.

Suppose that you have submitted a job with the Retry option set to 2 in the SCHEDULE command line argument. The master server starts the report request on the second slave engine on the NT-2 server. However, the NT-2 server runs out of temporary space and the job terminates. The master server will resubmit the job. Assuming that no other jobs have been submitted, this job is assigned to the first engine on the SUN-1 server.

The retry option is useful for giving you fail-over support, but should be used with caution. For example, setting the retry to a large number may not solve the problem. The resubmitted job may always fail if the underlying problem is with the report itself, not the engine.
6.2.5 Adding another slave server to the master

You want to add another slave server to the existing cluster configuration as defined in the following table:

<table>
<thead>
<tr>
<th>Machine/Server TNS name</th>
<th>Description</th>
<th>Master/Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN-2</td>
<td>4 CPU Sun Solaris server</td>
<td>Slave</td>
</tr>
</tbody>
</table>

*Note:* This exercise assumes that this machine has already been configured as a Reports Server. The TNS service entry name for the Reports Server is the machine name.

On the SUN-2 server machine (slave):

Open the `sun-2.ora` (the Reports Server configuration file) located in the `ORACLE_HOME\REPORT60\SERVER` directory and add the following:

```
nt-1.world=(ADDRESS=(PROTOCOL=tcp)(HOST=nt-1)(PORT=1949))
```

where `nt-1.world` is the name of the server instance and `.world` is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the `sqlnet.ora` file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the `sqlnet.ora`, then omit `.world` from the name of the server instance.

On the NT-1 server machine (master):

1. Open the `tnsnames.ora` file located in the `ORACLE_HOME\NET80\ADMIN` directory and add the following entry:

```
sun-2.world=(ADDRESS=(PROTOCOL=tcp)(HOST=sun-1)(PORT=1949))
```

where `sun-2.world` is the name of the server instance and `.world` is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the `sqlnet.ora` file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the `sqlnet.ora`, then omit `.world` from the name of the server instance.
2. Open the `nt-1.ora` (Reports Server configuration file) and add the following bolded text to the already existing CLUSTERCONFIG parameter:

```ora
clusterconfig="(server=nt-2
minengine=0
maxengine=2
initengine=2
cachedir="W:\Cache")
(server=sun-1
minengine=0
maxengine=2
initengine=2
cachedir="/share/Cache")
(server=sun-2
minengine=0
maxengine=4
initengine=4
cachedir="/share/Cache")"
```

3. Shut down and restart the master server so that the master server can recognize the newly configured slave server.

Suppose that while you were configuring the SUN-2 machine as a slave server, another administrator took down the NT-2 machine (for example, to perform a backup). While the NT-2 machine is still down, you restarted the Reports Server on the NT-1 machine. The NT-1 machine was able to start the slave engines on the two Sun machines, but could not start the slave engines on the NT-2 machine since it was down.

Since the NT-1 server is polling all the slave servers, once the NT-2 machine is brought back up and the Reports Server started, it will be detected automatically by the NT-1 server. When the four slave engines start, they are available to receive jobs from the master.
Customizing Reports at Runtime

Oracle Reports Developer can execute report definitions built with XML tags and merge them with other report definitions. Previously, a report had to be built and saved in the Report Builder in order to be run by Oracle Reports Developer. Now you can build a report definition using XML tags. This XML report definition can be run by itself or applied to another report at runtime to customize the output for a particular audience.

Using XML report definitions you can:

- Apply customizations to reports at runtime without changing the original report. By creating and applying different XML report definitions, you can alter the report output on a per user or user group basis. The advantage of this scenario is that you can use the same report to generate different output depending upon the audience.

- Apply batch updates to existing reports. When you apply an XML report definition to another report, you have the option of saving the combined definition out to a file. As a result, you can use XML report definitions to make batch updates to your existing reports. The advantage of this is that you can quickly update a large number of reports without having to open each file in the Report Builder to manually make the changes.

- Create complete report definitions in XML. The advantage of this is that you can build reports on the fly without using the Report Builder. If you can generate XML tags, you can create a report definition that can be run by Oracle Reports Developer.
7.1 Overview

Using XML tags, you can build a full or partial report definition that can serve as either a customization file or a completely self-contained report. A full report definition specifies a complete data model and layout in XML and can be run separately or applied to another report to customize it. A partial definition can contain far less information and can only be used in conjunction with another report (that is, it cannot be run by itself).

A customization file is a report definition that is applied to an existing report (.RDF or .XML). As illustrated in the figure below, it can change certain characteristics of existing report objects, such as the field’s date format mask or background color. A customization file can also be used to add entirely new objects to another report. Customization files can be full or partial report definitions.
In order to be run by itself, an XML report must contain a full report definition. As shown in the figure below, a self-contained XML report is one that is run without being applied to another report.
7.1.1 How to create and use XML report definitions

The steps below outline the process of building and using XML report definitions:

1. Create a full or partial report definition using the XML tags described in Section 7.5, "XML Tag Reference". You can create this definition manually with an editor or you can create it programmatically. The following is a sample of a partial report definition:

   <report name="emp" DTDVersion="1.0">
   <layout>
     <section name="main">
       <field name="f_sal" source="sal" textColor="red"/>
       <field name="f_mgr" source="mgr" fontSize="18" font="Script"/>
       <field name="f_deptno" source="deptno" fontStyle="bold" fontEffect="underline"/>
     </section>
   </layout>
   </report>

   Tip: This sample would change the formatting characteristics of some fields when applied to another report. Note that this XML could not be run by itself because it does not contain a full report definition. It contains no data model definition and only a partial layout definition. In order to be run by itself, it would need to contain a complete data model and layout definition.

   For more information on this step, refer to Section 7.2, "Creating an XML Report Definition".

2. Store the XML report definition in a location that is accessible to the Reports Server.

3. Apply the XML report definition to another report (via the CUSTOMIZE command line argument or the PL/SQL built-in SRW.APPLY_DEFINITION) or run the XML report definition by itself (via the REPORT command line argument).

   For more information on this step, refer to Section 7.3, "Running XML Report Definitions".

---

1 Creating the definition programmatically would allow you to build up a report definition on the fly based on user input.

2 Note that you can also use XML report definitions with the Reports Runtime and Report Builder.
The remainder of this chapter describes in greater detail the steps for building and using XML report definitions, and includes a reference section for the XML tags used to build a definition.

7.2 Creating an XML Report Definition

The best way to understand how to build an XML report definition is to work our way up from just the required tags to a partial definition and, finally, to a complete definition (that is, one that does not require a .RDF file in order to be run). This section describes the following XML definitions:

- Section 7.2.1, "Required tags"

  Some XML tags are required regardless of whether you are building a partial or full report definition in XML. This XML report definition shows you the minimum set of XML tags that a report definition must have in order to be parsed correctly.

- Section 7.2.2, "Partial report definitions"

  This type of XML report definition contains less than a complete report definition. As a result, it can only be applied to another report as a customization file. It cannot be run by itself.

- Section 7.2.3, "Full report definitions"

  This type of XML report definition contains a complete report definition. As a result, it can be applied to an .RDF file or it can be run by itself.

7.2.1 Required tags

Every XML report definition, full or partial, must contain the following required tag pair:

```xml
<report></report>
```

For example, the following is the most minimal XML report definition possible:1

```xml
<report name="emp" DTDVersion="1.0">
</report>
```

---

1 It should be noted that this XML report definition would have a null effect if applied to another report because it contains nothing. It can be parsed because it has the needed tags, but it is only useful to look at this definition to see what are the required tags.
The <report> tag indicates the beginning of the report, its name, and the version of the Document Type Definition (DTD) file that is being used with this XML report definition.\(^1\) The </report> tag indicates the end of the report definition.

A full report definition requires both a data model and a layout and therefore also requires the following tags and their contents:

- <data></data>
- <layout></layout>

### 7.2.2 Partial report definitions

One of the chief uses of XML report definitions is to make modifications to another report at runtime. The XML report definition enables you to easily change the data model or formatting of another report at runtime, without permanently affecting the original report.\(^2\) The advantage of this is that it enables you to use a single report to serve multiple audiences. For example, you can build one .RDF file and apply different partial XML report definitions to it to customize it for different audiences. The XML report definition can be very simple, containing only a few tags to change the appearance of a few objects, or very complex, affecting every object in the report and possibly adding new objects.

To help you understand the kind of modifications possible in customization files, it is helpful to see some examples. The Building Reports manual contains descriptions of how to build several example reports using Report Builder. The finished .RDF files for these reports are located in the \texttt{ORACLE_HOME/TOOLS/DOC60/US/RBBR60} directory. For the purposes of this chapter, an XML report definition that modifies some of these reports has been placed in this directory with the .RDF files. The table that follows describes each of these XML report definitions in greater detail.

---

\(^1\) DTD files are what give XML tags their meanings. Oracle Reports Developer includes a DTD file that defines the XML tags that can be used in a report definition. For more information about the supported XML tags, refer to Section 7.5, "XML Tag Reference".

\(^2\) Note that it is possible to save the combined .RDF file and XML report definition as a new .RDF file. This technique will be discussed later in this chapter.
<table>
<thead>
<tr>
<th>XML file</th>
<th>.RDF file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cond.xml</td>
<td>cond.rdf</td>
<td><strong>cond.xml changes:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the format mask of F_trade_date to MM/DD/RR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the fill colors of F_MinCurrent_pricePersymbol and F_MaxCurrent_pricePersymbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>cond.xml adds:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- HTML in the report escapes to be inserted when generating HTML output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, refer to Section 7.2.2.1, “Formatting modifications example”.</td>
</tr>
<tr>
<td>temp.xml</td>
<td>temp.rdf</td>
<td><strong>temp.xml changes:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the field labels for F_high_365 and F_low_365</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>temp.xml adds:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- a formatting exception to F_p_e to highlight values greater than 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- a formatting exception to F_p_e1 to highlight values greater than 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, refer to Section 7.2.2.2, “Formatting exception example”.</td>
</tr>
<tr>
<td>sect.xml</td>
<td>sect.rdf</td>
<td><strong>sect.xml adds:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- program units to the report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- link destinations to the detail records in the main section of the report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- hyperlinks from the employee summary in the header section to the detail records in the main section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, refer to Section 7.2.2.3, “Program unit and hyperlink example”.</td>
</tr>
</tbody>
</table>
You can apply the XML customizations by running the .RDF files with one additional argument. For example:

```
rwrun60 userid=scott/tiger report=cond.rdf
     customize=e:\orant\tools\doc60\us\rbbr60\cond.xml
```

For more information, refer to Section 7.3, "Running XML Report Definitions".

Take a few moments to run these .RDF files with and without the customization file. In the next section, we will examine the XML used to achieve these modifications.

### 7.2.2.1 Formatting modifications example

The XML in cond.xml modifies some basic formatting characteristics of cond.rdf and adds some HTML code to be inserted at the beginning and end of the report when generating HTML/CSS output.

#### Tips on this example

- Note that in this case the name attribute on the <report> tag matches the name of the .RDF file. You could also use a different name, for example, condnew.
- The name attributes on the <field> and <section> tags match the names of fields and the section that exist in the .RDF file. As a result, the other attributes on the <field> tag will be applied to those existing fields in the main section of the layout defined in the .RDF file.
The code inside of the <customize> tag modifies the before and after report escapes. The beforeReportType property indicates that the contents of the before report escape are located in a file. The beforeReportValue property indicates the name of the file, header_example.html, and its path (you might need to change this path if the file is located elsewhere on your machine). The afterReportType property indicates that the contents of the second report escape are located in the afterReportValue property. Note the use of the <!CDATA[]]> tag around the HTML for the afterReportValue property. When using characters in your XML report definition that could be confused with XML tags, you should always enclose those segments in the <!CDATA[]]> tag.

header_example.html contains a reference to a graphic, orep.gif, this graphic must be located in the same path as the HTML generated from the report.

To see the effects of the code in the <customize> tag, you need to generate HTML output. This report’s output is best viewed with HTMLCSS output (DESFORMAT=HTMLCSS) and page streaming (PAGESTREAM=YES).

```xml
<report name="cond" DTDVersion="1.0">
<layout>
<section name="main">
  <field name="f_trade_date" source="trade_date" formatMask="MM/DD/RR"/>
  <field name="F_Mincurrent_pricePersymbol" source="Mincurrent_pricePersymbol" lineColor="black" fillColor="r100g50b50"/>
  <field name="F_Maxcurrent_pricePersymbol" source="Maxcurrent_pricePersymbol" lineColor="black" fillColor="r100g50b50"/>
</section>
</layout>
<customize>
<object name="videosales" type="REP_REPORT">
<properties>
  <property name="beforeReportType">File</property>
  <property name="beforeReportValue">
    d:\orant\tools\doc60\us\rhbr60\header_example.html
  </property>
  <property name="afterReportType">Text</property>
  <property name="afterReportValue">
    <![CDATA[
      <center>...
    ]]>[CDATA[
  </property>
</object>
</customize>
```
7.2.2.2 Formatting exception example

The XML in temp.xml adds formatting exceptions to two fields in temp.rdf.

Tips on this example

- Note the usage of the `<exception>` tag to define the formatting change. This formatting exception is only applied when the criteria defined by the `<condition>` tag is met.

- The `<object>` tags inside of the `<customize>` section enable you to change the labels of an existing field in the layout. Note that if you are creating a new field, you can specify the label using the label attribute of the `<field>` tag.

```xml
<report name="temp" DTDVersion="1.0">
  <layout>
    <section name="main">
      <field name="f_p_e" source="p_e" alignment="right" formatMask="NNN0.00">
        <exception textColor="red">
          <condition source="p_e" operator="gt" operand1="10"/>
        </exception>
      </field>
      <field name="f_p_e1" source="p_e" alignment="right" formatMask="NNN0.00">
        <exception textColor="blue">
          <condition source="p_e" operator="gt" operand1="10"/>
        </exception>
      </field>
    </section>
  </layout>
</report>
```
Creating an XML Report Definition

7.2.2.3 Program unit and hyperlink example

The XML in sect.xml adds two program units to sect.rdf and uses the program units to add a header section.

Tips on this example

- When the parameter form appears, you should enter 100 for the parameter.
- Note that the program units are created outside of the data model and layout, inside the <programUnits> tag.
- The functions are referenced by name from the formatTrigger attribute of the <field> tag.
- Notice the usage of the <![CDATA[]]> tag around the PL/SQL function. This is necessary because of the special characters used within the PL/SQL code.
- This report is best viewed in PDF. To generate PDF output, you could use the following command line:
  
rwrun60 userid=scott/tiger@nt805 report=sect.rdf customize=sect.xml
destype=file desformat=htmlcss desname=d:\sect.pdf

Open the .PDF file and roll your mouse over the values in the SSN column. Click a value to be taken to the details on that record.

```xml
<report name="sect" DTDVersion="1.0">
  <layout>
    <section name="header">
      <field name="F_ssn1" source="ssn1"
        formatTrigger="F_ssn1FormatTrigger"/>
    </section>
  </layout>
</report>
```
Creating an XML Report Definition

```xml
</section>
<section name="main">
  <field name="F_ssn"
    source="ssn"
    formatTrigger="F_ssnFormatTrigger"/>
</section>
</layout>
</programUnits>
</report>

7.2.2.4 Data model and formatting modifications example

The XML in ref.xml adds a new query to the data model of ref.rdf and adds a header section.

**Tips on this example**

- Note that this XML report definition can be run by itself or applied to ref.rdf. The reason it can be run by itself is that it has both a data model and a complete layout.
Another important point is the use of aliases in the SELECT statement. In general, it is a good idea to use aliases in your SELECT lists because it guarantees the name that will be assigned to the report column. If you do not use an alias, the name of the report column is defaulted and could be something different from the name you expect (for example, portid1 instead of portid). This becomes important when you must specify the source attribute of the <field> tag because you have to use the correct name of the source column.

Also notice the use of the <labelAttribute> tag. This tag defines the formatting for the field labels in the layout. Because it lies outside of the <field> tags, it applies to all of the labels in the tabular layout. If you wanted it to pertain to only one of the fields, you could place it inside of the <field></field> tag pair. Be aware that if there is both a global and local <labelAttribute>, the local one overrides the global one. For more information refer to Section 7.5.8, "<field>".

```
<report name="ref" DTDVersion="1.0">
  <data>
    <dataSource name="Q_summary">
      <select>
        select portid ports, locname locations from portdesc
      </select>
    </dataSource>
  </data>
  <layout>
    <section name="header">
      <tabular name="M_summary" template="corp2.tdf">
        <labelAttribute font="Arial"
                        fontSize="10"
                        fontStyle="bold"
                        textColor="white"/>
        <field name="F_ports"
              source="ports"
              label="Port IDs"
              font="Arial"
              fontSize="10"/>
        <field name="F_locations"
              source="locations"
              label="Port Names"
              font="Arial"
              fontSize="10"/>
      </tabular>
    </section>
  </layout>
</report>
```
7.2.3 Full report definitions

Another use of XML report definitions is to make an entire report definition in XML that can be run independently of another report. The advantage of this is that you can build a report without using the Report Builder. In fact, you could even use your own front end to generate the necessary XML and allow your users to build their own reports dynamically.

The following example illustrates a complete report definition in XML. This XML report definition is named videosales.xml and can be found in the ORACLE_HOME\TOOLS\DOC60\US\RBBR60 directory.

Tips on this example

- Note that this XML report definition is complete and can be run by itself. It contains a full data model and layout. This report is best viewed in PDF.

- The first query in the data model (Q_1) is used to populate a summary tabular layout in the header section of the report. The second query (Q_2) is used for the matrix break layout in the main section of the report. Note that the <group>, <matrixRow>, <matrixCol>, and <matrixCell> tags define both the layout and the data model structure needed to support it. Based on which fields are inside these tags, the groups and columns are arranged within the data model. To get a better sense of the data model, you can run the report to the Report Builder and look at the Data Model view of the Report Editor:

  rwbld60 userid=scott/tiger report=videosales.xml

- The quarter and city values in the header section are hyperlinked to the quarter and city values in the main section. This is accomplished by associating format triggers with each of the fields that contain quarter and city values. The PL/SQL for the triggers is located inside the <programUnits> tag at the end of the report definition. When the report is used to generate PDF or HTMLCSS output, the user can click on values in the summary in the header section to jump to the details in the main section of the report.

  <report name="videosales" author="Generated" DTDVersion="1.0">
  <data>
  <dataSource name="Q_1">
  <select>
  SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER,
  VIDEO_CATEGORY_BY_QTR.SALES_REGION,
  VIDEO_CATEGORY_BY_QTR.STATE, VIDEO_CATEGORY_BY_QTR.CITY,
  VIDEO_CATEGORY_BY_QTR.PRODUCT_CATEGORY,
  VIDEO_CATEGORY_BY_QTR.TOTAL_SALES,
<data>
  <select>
    SELECT VIDEO_CATEGORY_BY_QTR.QUARTER,
    VIDEO_CATEGORY_BY_QTR.CITY,
    VIDEO_CATEGORY_BY_QTR.PRODUCT_CATEGORY,
    VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT,
    VIDEO_CATEGORY_BY_QTR.TOTAL_SALES,
    VIDEO_CATEGORY_BY_QTR.TOTAL_COST
    FROM SCOTT.VIDEO_CATEGORY_BY_QTR
    WHERE VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
  </select>
</dataSource>

<summary name="SumTOTAL_SALESPerCITY1" source="total_sales1"/>
<summary name="SumTOTAL_COSTPerCITY1" source="total_cost1"/>
<summary name="SumTOTAL_PROFITPerCITY1" source="total_profit1"/>
<summary name="SumTOTAL_SALESPerQUARTER" source="total_sales"/>
<summary name="SumTOTAL_COSTPerQUARTER" source="total_cost"/>
<summary name="SumTOTAL_PROFITPerQUARTER" source="total_profit"/>
<summary name="SumTOTAL_SALESPerCITY" source="total_sales"/>
<summary name="SumTOTAL_COSTPerCITY" source="total_cost"/>
<summary name="SumTOTAL_PROFITPerCITY" source="total_profit"/>
<formula name="Profit_Margin" source="FormulaProfitMargin"
  datatype="number" width="9"/>
</data>
</layout>

<section name="header">
  <groupLeft name="M_video_sales_summary" template="corp1.tdf">
    <group>
      <field name="f_quarter1" source="quarter1" label="Quarter"
        font="Arial" fontSize="8"
        formatTrigger="F_quarter1FormatTrigger">
        <labelAttribute font="Arial" fontSize="8"
          fontStyle="bold" textColor="yellow"/>
      </field>
    </group>
  </groupLeft>
</section>
Creating an XML Report Definition

```
<field name="SumTOTAL_SALESPerCITY1" source="SumTOTAL_SALESPerCITY1"
      label="Sales" font="Arial" fontSize="8"
      formatMask="LNNNGNNNGNNNGNN0D00">
  <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
</field>

<field name="f_SumTOTAL_COSTPerCITY1" source="SumTOTAL_COSTPerCITY1"
      label="Costs" font="Arial" fontSize="8"
      formatMask="LNNNGNNGNNGNNNN0D00">
  <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
</field>

<field name="f_SumTOTAL_PROFITPerCITY1" source="SumTOTAL_PROFITPerCITY1"
      label="Profits" font="Arial" fontSize="8"
      formatMask="LNNNGNNNGNNNGNN0D00">
  <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
</field>

<field name="Profit_Margin" source="Profit_Margin"
      label="Margin%" font="Arial" fontSize="8"
      formatMask="N0">%
  <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
</field>
```

```
<group>
  <field name="f_quarter" source="quarter" label="Quarter:"
    font="Arial" fontSize="8"
    formatTrigger="F_quarterFormatTrigger">
    <labelAttribute font="Arial" fontSize="8"
        fontStyle="bold" textColor="black"/>
  </field>

  <field name="f_SumTOTAL_SALESPerQUARTER" source="SumTOTAL_SALESPerQUARTER"
    label="Qtrly: Sales: " font="Arial" fontSize="8"
```
Creating an XML Report Definition

Customizing Reports at Runtime  7-101
<field name="f_SumTOTAL_PROFITPerCITY"
source="SumTOTAL_PROFITPerCITY"
label="Sales: " font="Arial" fontSize="8" fontStyle="bold" textColor="yellow" formatMask="LNNNGNNNGNNNGNN0D00">
</field>

<matrixCol>
<matrixRow name="g_product_category">
<field name="f_product_category" source="product_category"
label="Product Category" font="Arial" fontSize="8"/>
</matrixRow>
<matrixCell name="g_total_sales">
<field name="f_total_sales" source="total_sales" label="Total Sales"
font="Arial" fontSize="8" lineColor="noLine" formatMask="LNNNGNNNGNNNGNN0D00"/>
<field name="f_total_cost" source="total_cost" label="Total Cost"
font="Arial" fontSize="8" lineColor="noLine" formatMask="LNNNGNNNGNNNGNN0D00"/>
<field name="f_total_profit" source="total_profit" label="Total Profit"
font="Arial" fontSize="8" lineColor="noLine" formatMask="LNNNGNNNGNNNGNN0D00"/>
</matrixCell>
</matrix>
</section>
</layout>
<programUnits>
<function name="F_quarter1FormatTrigger">
<![CDATA[
function F_quarter1FormatTrigger return boolean is
begin
SRW.SET_HYPERLINK('"QUARTER_DETAILS_&<"' || LTRIM(:quarter1) || '">');
return (TRUE);
end;
]]>
</function>
<function name="F_quarterFormatTrigger">
<![CDATA[
function F_quarterFormatTrigger return boolean is
begin
SRW.SET_LINKTAG('"QUARTER_DETAILS_&<"' || LTRIM(:quarter) || '">');
return (TRUE);
end;
]]>
</function>
    return (TRUE);
    end;
  ]]>>
</function>

<function name="F_cityFormatTrigger">
  <![CDATA[
    function F_cityFormatTrigger return boolean is
    begin
      SRW.SET_LINKTAG('QTR_CITY_DETAILS_' || LTRIM(:quarter) ||
      LTRIM(:city) || '');</
      return (TRUE);
    end;
  ]]>>
</function>

<function name="FormulaProfitMargin">
  <![CDATA[
    FUNCTION FormulaProfitMargin RETURN number IS
    BEGIN
      return ((:TOTAL_PROFIT1 / (:TOTAL_SALES1 - (0.07 * :TOTAL_SALES1)))
      * 100); END;
  ]]>>
</function>
</programUnits>
</report>
7.3 Running XML Report Definitions

Once you have created your XML report definition, you can use it in the following ways.

- Section 7.3.1, "Applying an XML report definition at runtime" You can apply XML report definitions to .RDF or other .XML files at runtime by specifying the CUSTOMIZE command line argument or the SRW.APPLY DEFINITION built-in.

- Section 7.3.2, "Running an XML report definition by itself" You can run an XML report definition by itself (without another report) by specifying the REPORT command line argument.

- Section 7.3.3, "Performing batch modifications" You can use RWCON60 to make batch modifications using the CUSTOMIZE command line argument.

The sections that follow describe each of the above cases in more detail and provide examples.

7.3.1 Applying an XML report definition at runtime

To apply an XML report definition to an .RDF or .XML file at runtime, you can use the CUSTOMIZE command line argument or the SRW.APPLY DEFINITION built-in. CUSTOMIZE can be used with RW60CLI, RWRUN60, RWBLD60, RWCON60, and URL report requests. For more information on using CUSTOMIZE with RWCON60, refer to Section 7.3.3, "Performing batch modifications".

7.3.1.1 Applying one XML report definition

The following command line would send a job request to the Reports Server that applies an XML report definition, emp.xml, to an .RDF file, emp.rdf:

```
rwcli60 report=emp.rdf customize=e:\myreports\emp.xml
    userid=username/password@mydb desextension=file desname=emp.pdf desformat=PDF
    server=repserver
```

If you were using Reports Runtime, the equivalent command line would be:

```
rwrut60 userid=username/password@mydb report=emp.rdf
    customize=e:\myreports\emp.xml desextension=file desname=emp.pdf desformat=PDF
```
Tip: When testing your XML report definition, it is sometimes useful to run your report requests with additional arguments to create a trace file. For example:

```
tracefile=emp.log tracemode=trace_replace traceopt=trace_app
```

The trace file provides a detailed listing of the creation and formatting of the report objects.

### 7.3.1.2 Applying multiple XML report definitions

You can apply multiple XML report definitions to a report at runtime by providing a list with the CUSTOMIZE command line argument. The following command line would send a job request to the Reports Server that applies two XML report definitions, emp0.xml and emp1.xml, to an .RDF file, emp.rdf:

```
rwcli60 report=emp.rdf
    customize="(e:\corp\myreports\emp0.xml,
                e:\corp\myreports\empl.xml)"
    userid=username/password@mydb destype=file desname=emp.pdf desformat=PDF
    server=repserver
```

If you were using Reports Runtime, the equivalent command line would be:

```
rwrun60 report=emp.rdf
    customize="(e:\corp\myreports\emp0.xml,
                e:\corp\myreports\empl.xml)"
    userid=username/password@mydb destype=file desname=emp.pdf desformat=PDF
```

### 7.3.1.3 Applying an XML report definition in PL/SQL

To apply an XML report definition to an .RDF file in PL/SQL, you use the SRW.APPLY_DEFINITION and SRW.ADD_DEFINITION built-ins in the Before Form or After Form trigger.

#### 7.3.1.3.1 Applying an XML definition stored in a file

To apply XML that is stored in the file system to a report, you can use the SRW.APPLY_DEFINITION built-in in the Before Form or After Form triggers of the report:

```
SRW.APPLY_DEFINITION ('d:\orant\tools\doc60\us\rbbr60\cond.xml');
```

When the report is run, the trigger will execute and the specified XML file will be applied to the report.
7.3.1.3.2 Applying an XML definition stored in memory

To create an XML report definition in memory, you must add the definition to the document buffer using SRW.ADD_DEFINITION before applying it using SRW.APPLY_DEFINITION.

The following example illustrates how to build up several definitions in memory based upon parameter values entered by the user and then apply them. The PL/SQL in this example is actually used in the After Parameter Form trigger of an example report called videosales_custom.rdf that can be found in the ORACLE_HOME\TOOLS\DOC60\US\RBBR60 directory.

videosales_custom.rdf contains PL/SQL in its After Parameter Form trigger that does the following:

- conditionally highlights fields based upon parameter values entered by the user at runtime
- changes number format masks based upon parameter values entered by the user at runtime

Tips on this example

- Note that each time you use SRW.APPLY_DEFINITION, the document buffer is flushed and you must begin building a new XML report definition with SRW.ADD_DEFINITION.
- Notice the use of the parameters hilite_profits, hilite_costs, hilite_sales, and money_format to determine what to include in the XML report definition. hilite_profits, hilite_costs, and hilite_sales are also used in the formatting exceptions to determine which values to highlight.
- Because of the upper limit on the size of VARCHAR2 columns, you may need to spread very large XML report definitions across several columns. If so, you might have to create several definitions in memory and apply them separately rather than creating one large definition and applying it once.

```plsql
function AfterPForm return boolean is
begin
  SRW.ADD_DEFINITION('<report name="vidsales_masks"
    author="Generated" DTDVersion="1.0"/>');
  IF :MONEY_FORMAT='$$NNN.00' THEN
    SRW.ADD_DEFINITION(' <layout/>
    SRW.ADD_DEFINITION('  <section name="main">
    SRW.ADD_DEFINITION('   <field name="F_TOTAL_PROFIT"
      source="TOTAL_PROFIT" formatMask="LNNNNNNNNNNN0D00"/>
    SRW.ADD_DEFINITION('   <field name="F_TOTAL_SALES"
      source="TOTAL_SALES" formatMask="LNNNNNNNNNNN0D00"/>
  );
end AfterPForm;```
SRW.ADD_DEFINITION('   <field name="F_TOTAL_COST" source="TOTAL_COST" formatMask="LNNNNNNNNNNN0D00"/>');
SRW.ADD_DEFINITION('   <field name="F_SumTOTAL_PROFITPerCITY" source="SumTOTAL_PROFITPerCITY" formatMask="LNNNNNNNNNNN0D00"/>');
SRW.ADD_DEFINITION('   <field name="F_SumTOTAL_SALESPerCITY" source="SumTOTAL_SALESPerCITY" formatMask="LNNNNNNNNNNN0D00"/>');
SRW.ADD_DEFINITION('   <field name="F_SumTOTAL_COSTPerCITY" source="SumTOTAL_COSTPerCITY" formatMask="LNNNNNNNNNNN0D00"/>');
SRW.ADD_DEFINITION('  </section>');
SRW.ADD_DEFINITION(' </layout>');

ELSIF :MONEY_FORMAT='"NNNN' THEN
SRW.ADD_DEFINITION(' <layout>');
SRW.ADD_DEFINITION('  <section name="main">');
SRW.ADD_DEFINITION('   <field name="F_TOTAL_PROFIT" source="TOTAL_PROFIT">');
SRW.ADD_DEFINITION('   <field name="F_TOTAL_SALES" source="TOTAL_SALES">');
SRW.ADD_DEFINITION('   <field name="F_TOTAL_COST" source="TOTAL_COST">');
SRW.ADD_DEFINITION('   <field name="F_SumTOTAL_PROFITPerCITY" source="SumTOTAL_PROFITPerCITY">');
SRW.ADD_DEFINITION('   <field name="F_SumTOTAL_SALESPerCITY" source="SumTOTAL_SALESPerCITY">');
SRW.ADD_DEFINITION('   <field name="F_SumTOTAL_COSTPerCITY" source="SumTOTAL_COSTPerCITY">');
SRW.ADD_DEFINITION('  </section>');
SRW.ADD_DEFINITION(' </layout>');

END IF;
SRW.ADD_DEFINITION('</report>');
SRW.APPLY DEFINITION;
SRW.ADD_DEFINITION('<report name="vidsales_hilite_costs" author="Generated" DTDVersion="1.0">');

IF :HILITE_COSTS <> 'None' THEN
SRW.ADD_DEFINITION(' <section name="main">');
SRW.ADD_DEFINITION('   <field name="F_SumTOTAL_COSTPerCITY" source="SumTOTAL_COSTPerCITY" formatMask="LNNNNNNNNNNN0D00"/>');
SRW.ADD_DEFINITION('   <excepton textColor="red">');
SRW.ADD_DEFINITION('     <condition source="TOTAL_COST" operator="gt" operand1":"HILITE_COSTS"/>');
SRW.ADD_DEFINITION('   </exception>');
SRW.ADD_DEFINITION(' </section>');
SRW.ADD_DEFINITION(' </layout>');

END IF;

SRW.ADD_DEFINITION('</report>');
Running XML Report Definitions

SRW.ADD_DEFINITION(' </exception>');
SRW.ADD_DEFINITION(' </field>');
SRW.ADD_DEFINITION(' </section>');
SRW.ADD_DEFINITION(' </layout>');
END IF;
SRW.ADD_DEFINITION( </report>');
SRW.APPLY_DEFINITION;
SRW.ADD_DEFINITION('<report name="vidsales_hilite_sales"
    author="Generated" DTDVersion="1.0">');
IF :HILITE_SALES <> 'None' THEN
  SRW.ADD_DEFINITION(' <layout>');
  SRW.ADD_DEFINITION('   <section name="main">');
  SRW.ADD_DEFINITION('     <field name="F_TOTAL_SALES"
        source="TOTAL_SALES">');
  SRW.ADD_DEFINITION('     <exception textColor="red">');
  SRW.ADD_DEFINITION('       <condition source="TOTAL_SALES"
            operator="gt" operand1=":hilite_sales"/>');
  SRW.ADD_DEFINITION('     </exception>');
  SRW.ADD_DEFINITION('   </field>');
  SRW.ADD_DEFINITION(' </section>');
  SRW.ADD_DEFINITION(' </layout>');
END IF;
SRW.ADD_DEFINITION( </report>');
SRW.APPLY_DEFINITION;
SRW.ADD_DEFINITION('<report name="vidsales_hilite_profits"
    author="Generated" DTDVersion="1.0">');
IF :HILITE_PROFITS <> 'None' THEN
  SRW.ADD_DEFINITION(' <layout>');
  SRW.ADD_DEFINITION('   <section name="main">');
  SRW.ADD_DEFINITION('     <field name="F_TOTAL_PROFIT"
        source="TOTAL_PROFIT">');
  SRW.ADD_DEFINITION('     <exception textColor="red">');
  SRW.ADD_DEFINITION('       <condition source="TOTAL_PROFIT"
            operator="gt" operand1=":hilite_profits"/>');
  SRW.ADD_DEFINITION('     </exception>');
  SRW.ADD_DEFINITION(' </section>');
  SRW.ADD_DEFINITION(' </layout>');
END IF;
SRW.ADD_DEFINITION( </report>');
SRW.APPLY_DEFINITION;
return (TRUE);
end;
7.3.2 Running an XML report definition by itself

To run an XML report definition by itself, you send a request with an XML file specified in the REPORT argument. The following command line sends a job request to the Reports Server to run a report, emp.xml, by itself:

```
rwcli60 userid=username/password@mydb
    report=e:\corp\myreports\emp.xml
    desctype=file desname=emp.pdf desformat=PDF
    server=repserver
```

If you were using Reports Runtime, the equivalent command line would be:

```
rwrn60 userid=username/password@mydb
    report=e:\corp\myreports\emp.xml
    desctype=file desname=emp.pdf desformat=PDF
```

Tip: When running an XML report definition in this way, the file extension must be .XML. Note also that you could apply an XML customization file to this report using the CUSTOMIZE argument.

7.3.3 Performing batch modifications

As illustrated in the figure below, if you have a large number of reports that need to be updated, you can use the CUSTOMIZE command line argument with RWCON60 to perform modifications in batch. Batch modifications are particularly useful when you need to make a repetitive change to a large number of reports (for example, changing a field’s format mask). Rather than opening each report and manually making the change in Report Builder, you can run RWCON60 once and make the same change to a large number of reports at once.
The following example applies two XML report definitions, translate.xml and customize.xml, to three .RDF files, inven.rdf, inven2.rdf, and manu.rdf, and saves the revised definitions to new files, inven1_new.rdf, inven2_new.rdf, and manu_new.rdf.

```
rwcon60 username/password@mydb
  stype=rdfFile
  source="(inven1.rdf, inven2.rdf, manu.rdf)"
  dtype=rdfFile
  dest="(inven1_new.rdf, inven2_new.rdf, manu_new.rdf)"
  customize="(e:\apps\trans\translate.xml,
e:\apps\custom\customize.xml)" batch=yes
```

### 7.4 Debugging XML Report Definitions

The following features can help you to debug your XML report definitions:

- Section 7.4.1, "XML Parser error messages"
- Section 7.4.2, "Tracing options"
- Section 7.4.3, "RWBLD60"
- Section 7.4.4, "TEXT_IO"
7.4.1 XML Parser error messages

The XML parser will catch most syntax errors and display an error message. The error message contains the line number in the XML where the error occurred as well as a brief description of the problem.

7.4.2 Tracing options

When testing your XML report definition, it is sometimes useful to run your report requests with additional arguments to create a trace file. For example:

```
rwrun60 username/password@mydb
  report=e:\corp\myreports\emp.xml
  tracefile=emp.log
  tracemode=trace_replace
  traceopt=trace_app
```

The last three arguments in this command line will generate a trace file that provides a detailed listing of the fetching and formatting of the report. Below is a segment of an example trace file for a successfully executed report.

```
LOG :
  Report: d:\xml_reps\test1.xml
  Logged onto server:
  Username:

LOG :
  Logged onto server: nt805
  Username: scott

+----------------------------------------+
| Report customization/generation begins |
+----------------------------------------+

Processing XML report definition 1 of 1.
  *** Parsing the XML document ***
  Creating XML parser object...
  XML Parser Created!
  Parsing report definition from:
  d:\xml_reps\test1.xml
  Report definition parsed successfully!
  *** Setting Application Property ***
  Setting module name to "test"...
  Done with application level properties modification.
  *** Creating PL/SQL Program Units ***
  *** Defaulting the Data Model ***
  Created query Q Depemp.
  Applying SQL to query Q Depemp and creating columns...
```
Done with queries and columns creation/modification.
Done with groups creation/modification.
*** Defaulting the Layout ***
Start defaulting layout for main section...
Defaulting field f_deptno for column deptno...
Defaulting field f_mgr for column mgr...
Defaulting field f_job for column job...
Layout defaulted into new frame M_empform.
*** Modifying report objects’ properties ***
+-------------------------------------------------------+
| Report customization/generation finished successfully |
+-------------------------------------------------------+

11:22:59 APP ( Frame
11:22:59 APP . ( Text Boilerplate B_DATE1_SEC2
11:22:59 APP . ) Text Boilerplate B_DATE1_SEC2
11:22:59 APP . ( Text Boilerplate B_PAGENUM1_SEC2
11:22:59 APP . ) Text Boilerplate B_PAGENUM1_SEC2
11:22:59 APP . ( Text Field F_DATE1_SEC2
11:22:59 APP .. ( Database Column Name unknown
11:22:59 APP .. ) Database Column Name unknown
11:22:59 APP . ) Text Field F_DATE1_SEC2
11:22:59 APP ) Frame
11:22:59 APP ( Frame
11:22:59 APP . ( Frame M_G_1_GRPFR
11:22:59 APP . ( Frame M_G_1_HDR
11:22:59 APP ... ( Text Boilerplate B_DEPTNO
11:22:59 APP ... ) Text Boilerplate B_DEPTNO
11:22:59 APP ... ( Text Boilerplate B_MGR
11:22:59 APP ... ) Text Boilerplate B_MGR
11:22:59 APP ... ( Text Boilerplate B_JOB
11:22:59 APP ... ) Text Boilerplate B_JOB
11:22:59 APP .. ) Frame M_G_1_HDR
11:22:59 APP .. ( Repeating Frame R_G_1
11:22:59 APP ... ( Group G_1 Local Break: 0 Global Break: 0
11:22:59 APP .... ( Query Q_depemp
11:22:59 SQL EXECUTE QUERY : select * from emp
11:22:59 APP .... ) Query Q_depemp
11:22:59 APP ... ) Group G_1
11:22:59 APP ... ( Text Field F_DEPTNO
11:22:59 APP ... ( Database Column DEPTNO
11:22:59 APP ... ) Database Column DEPTNO

7-112 Oracle Reports Developer Publishing Reports
7.4.3 RWBLD60

When designing an XML report definition, it is sometimes useful to open it in Report Builder. In Report Builder, you can quickly determine if the objects are being created or modified as expected. For example, if you are creating summaries in an XML report definition, opening the definition in Report Builder enables you to quickly determine if the summaries are being placed in the appropriate group in the data model.

To open a full report definition in Report Builder, you use the REPORT keyword. For example:

```
  rwbld60 userid=username/password@mydb
  report=e:\corp\myreports\emp.xml
```

To open a partial report definition in Report Builder, you use the CUSTOMIZE keyword. For example:

```
  rwbld60 userid=username/password@mydb report=emp.rdf
  customize=e:\myreports\emp.xml
```

In both cases, the Report Builder is opened with the XML report definition in effect. You can then use the various views (including the Live Previewer) of the Report Editor to quickly determine if the report is being created or modified as you expected.
7.4.4 TEXT_IO

If you are using SRW.ADD_DEFINITION to build an XML report definition in memory, it can be helpful to write the XML to a file for debugging purposes. Following is an example of a procedure that writes each line that you pass it to the document buffer in memory and, optionally, to a file that you give it.

```plsql
PROCEDURE addaline (newline VARCHAR, outfile Text_IO.File_Type) IS
BEGIN
    SRW.ADD_DEFINITION(newline);
    IF :WRITE_TO_FILE='Yes' THEN
        Text_IO.Put_Line(outfile, newline);
    END IF;
END;
```

Note that for this example to work, the PL/SQL that calls this procedure would need to declare a variable of type TEXT_IO.File_Type. For example:

```plsql
custom_summary  Text_IO.File_Type;
```

And you would also need to open the file for writing and call the addaline procedure, passing it the string to be written and the file to which it should be written. For example:

```plsql
custom_summary := Text_IO.Fopen(:file_directory || 'vid_summ_per.xml', 'w');
addaline('report name="video_custom" author="Generated" DTDVersion="1.0">',
custom_summary);
```

7.5 XML Tag Reference

The Document Type Definition (DTD) file incorporated into Oracle Reports Developer defines the tags that can be used in an XML report definition. The sections that follow describe each of the tags and their syntax, and provide examples of their usage. Note that the tags are listed in hierarchical order (from outermost to innermost).

**Warning:** The XML tags and their attributes are case sensitive, and should be entered in the case shown in the syntax descriptions.
### 7.5.1 <!-- comments -->

**Description:** The <!-- --> tag enables you to include comments within your XML report definition. The parser will ignore any text between the comment delimiters.

**Tip:** If you are using PL/SQL (SRW.ADD_DEFINITION) to build your XML report definition, you can incorporate comments in the program unit using the PL/SQL comment delimiters (for example, -- or /* */).

**Syntax:**
```
<!--
comment_content
-->
```

**Examples:** The following example shows a segment of an XML report definition that uses the <!-- --> tag to include a comment.

```xml
<report name="cond" DTDVersion="1.0">
<!-- This report assumes that the file named header_example.html is located in d:\ORANT\TOOLS\DOC60\US\RBBR60. If it it not located there, the report will not run properly. -->
```

### 7.5.2 <![CDATA[]]>

**Description:** The <![CDATA[]]> tag enables you to include special characters within your XML report definition. The parser will ignore any special characters it encounters within the <![CDATA[]]> tag. This is particularly useful when including PL/SQL program units or SQL queries that may require special characters.

**Syntax:**
```
<![CDATA[
content
]]>
```

**Examples:** The following example shows a segment of an XML report definition that uses the <![CDATA[]]> tag to protect a PL/SQL function that adds a hyperlink and hyperlink destination to an object in a report.

```xml
<programUnits>
  <function name="F_ssn1FormatTrigger">
    <![CDATA[
      function F_ssn1FormatTrigger return boolean is
      begin
        SRW.SET_HYPERlink('#EMP_DETAILS_&<\ TTRIM(TO_CHAR(:SSN)) || '}>');
        return (TRUE);
    end;]]>
```
The following example shows a segment of an XML report definition that uses the <![CDATA[]]> tag to protect a SQL statement that contains a greater than sign.

```
<select>
  <![CDATA[
    SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER,
    VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT
    FROM SCOTT.VIDEO_CATEGORY_BY_QTR
    WHERE (VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
      AND VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT>2000)
  ]]>}
</select>
```

### 7.5.3 <condition>

**Description:** The <condition> tag defines the conditions under which a formatting exception is applied to a field. The <condition> tag must be nested within an <exception> tag.

For more information refer to Section 7.5.7, "<exception>"
Attributes: The following table describes the attributes of the `<condition>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>required</td>
<td>Is the name of the source column to be used in the condition.</td>
</tr>
<tr>
<td>operator</td>
<td>required</td>
<td>Is the operator to use in comparing other values to the source column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- eq (equal)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- lt (less than)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- lteq (less than or equal)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- neq (not equal)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- gt (greater than)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- gteq (greater than or equal)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- btw (between)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- notBtw (not between)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- like</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- notLike</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- null</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- notNull</td>
</tr>
<tr>
<td>operand1</td>
<td>optional</td>
<td>Is the value to which the source column is being compared. If the operator is null or notNull, no operands are required. If the operator is btw or notBtw, then you must also specify operand2.</td>
</tr>
<tr>
<td>operand2</td>
<td>optional</td>
<td>Is the second value to which the source column is being compared. You only need to use operand2 if the operator requires two values for comparison (that is, if the operator is btw or notBtw)</td>
</tr>
<tr>
<td>relation</td>
<td>optional</td>
<td>Defines whether there are multiple conditions and, if there are, how they should be related.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- and means that the formatting exception is applied only if both are met.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- or means that the formatting exception is applied if either condition is met.</td>
</tr>
</tbody>
</table>
Usage Notes: Two conditions can be joined by entering the relation attribute in the first condition tag, which must include either of the operators and or or.

Example: The following example shows two formatting exceptions for field f_ename. The first exception changes the text color to red if both of its conditions are met. The second exception changes the text color to blue if its condition is met.

```xml
<field name="f_ename" source="ename" label="Employee" textColor="green">
    <exception textColor="red">
        <condition source="deptno" operator="btw" operand1="20" operand2="30" relation="and"/>
        <condition source="sal" operator="gt" operand1="1000"/>
    </exception>
    <exception textColor="blue">
        <condition source="deptno" operator="eq" operand1="30"/>
    </exception>
</field>
```

7.5.4 <customize>

Description: The <customize> tag delimits any object properties that you want to specify as part of the report definition. The tags nested within the <customize> tag (<object> <properties>, and <property>) enable you to set properties for certain objects in the report.

Syntax:
```
<customize>
    content_of_data_model
</customize>
```

Example: The following example shows the object property segment of an XML report definition.

```xml
<customize>
    <object name="videosales" type="REP_REPORT">
        <properties>
            <property name="beforeReportType">File</property>
            <property name="beforeReportValue">
                d:\xml_reps\header_example.html
            </property>
            <property name="afterReportType">Text</property>
            <property name="afterReportValue">
                <![CDATA[
                    <center>
                        <font face="Arial,Helvetica"><font size=-1><font color="#000000">
                            Send questions to <a href="mailto:your_email_id">YourNameHere</a>.
                        </font></font></center>
                ]]>[CDATA[
            </property>
        </properties>
    </object>
</customize>
```
The following example shows a segment of an XML report definition that changes some boilerplate text. This is useful for changing labels for existing fields.

```
<customize>
  <object name="B_high_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">High</property>
    </properties>
  </object>
  <object name="B_low_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">Low</property>
    </properties>
  </object>
</customize>
```

7.5.5 <data>

**Description:** The `<data>` tag delimits the beginning and ending of the data model of the report definition.

**Syntax:**
```
<data>
  content_of_data_model
</data>
```

**Example:** The following example shows the data model segment of an XML report definition.

```
<data>
  <dataSource name="q_category">
    <select>
      SELECT ic.category,
            SUM (h.sales),
            AVG (h.high_365),
            AVG (h.low_365),
            AVG (h.div),
            AVG (h.p_e)...
    </select>
  </dataSource>
</data>
```
FROM stock_history h, indcat ic
WHERE h.symbol=ic.symbol
GROUP BY ic.category
</select>
</dataSource>
</data>

The following example shows a segment of an XML report definition that uses the
<![CDATA[]]> tag to protect a SQL statement that contains a greater than sign.

<data>
<dataSource name="Q_1">
<select>
<![CDATA[
SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER,
    VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT
FROM SCOTT.VIDEO_CATEGORY_BY_QTR
WHERE (VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
    AND VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT>2000)
]]>}
</select>
</dataSource>
</data>

7.5.6 <dataSource>
Description: The <dataSource> tag delimits the beginning and ending of a query in the data
model. <dataSource> must be nested within the <data> tag. All of the data sources
supported by Oracle Reports Developer (SQL and Express) are supported by this
tag.

Syntax: <dataSource>
    content_of_data_source
</dataSource>

Example: The following example shows the data model segment of an XML report definition.

<data>
    <dataSource name="q_category">
        <select>
            SELECT ic.category,
                SUM (h.sales),
                AVG (h.high_365),
                AVG (h.low_365),
                AVG (h.div),
                AVG (h.p_e)
FROM stock_history h, indcat ic
WHERE h.symbol=ic.symbol
GROUP BY ic.category
</select>
</dataSource>
</data>

The following example shows a segment of an XML report definition that uses the
<![CDATA[]]> tag to protect a SQL statement that contains a greater than sign.

<data>
<dataSource name="Q_1">
<select>
<![CDATA[
SELECT ALL VIDEO_CATEGORY_BY_QTR QUARTER,
    VIDEO_CATEGORY_BY_QTR TOTAL_PROFIT
FROM SCOTT VIDEO_CATEGORY_BY_QTR
WHERE (VIDEO_CATEGORY_BY_QTR SALES_REGION='West'
    AND VIDEO_CATEGORY_BY_QTR TOTAL_PROFIT>2000)
]]>
</select>
</dataSource>
</data>

7.5.7 <exception>

Description: The <exception> tag delimits a formatting exception that you want to apply to a
field (for example, the field should turn red when the value exceeds some limit). The
<exception> tag must be nested within a <field> tag. It must also have a
<condition> tag nested within it that defines the condition under which to apply
the formatting exception.

For more information refer to:
Section 7.5.8, "<field>"
Section 7.5.3, "<condition>"
Syntax:

```xml
<exception
    [lineColor="color_name | noLine"]
    [fillColor="color_name | noFill"]
    [textColor="color_name"]
    [hide="yes | no"]
    [font="font_name"]
>
    condition_definition
</exception>
```

Attributes:

The following table describes the attributes of the `<exception>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineColor</td>
<td>optional</td>
<td>Is the name of the border color to apply when the condition is met. If noLine is specified, the border will be transparent (that is, invisible).</td>
</tr>
<tr>
<td>fillColor</td>
<td>optional</td>
<td>Is the name of the fill color to apply when the condition is met. If noFill is specified, the background will be transparent.</td>
</tr>
<tr>
<td>textColor</td>
<td>optional</td>
<td>Is the name of the text color to apply when the condition is met.</td>
</tr>
<tr>
<td>hide</td>
<td>optional</td>
<td>Is whether to hide the field when the condition is met.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes means the field will be hidden when the condition is met.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no means the field will not be hidden when the condition is met.</td>
</tr>
<tr>
<td>font</td>
<td>optional</td>
<td>Is the name of the font to apply when the condition is met.</td>
</tr>
<tr>
<td>fontSize</td>
<td>optional</td>
<td>Is the size of the font to be used when the condition is met.</td>
</tr>
<tr>
<td>fontStyle</td>
<td>optional</td>
<td>Is the style of the font to be used when the condition is met:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• regular</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• italic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• bold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• boldItalic</td>
</tr>
</tbody>
</table>
Usage Notes:

- Exceptions are processed in the order they appear in the field.
- Each exception can have up to three conditions.
- There is no limit on the number of exceptions that can be applied to a field, except for the PL/SQL maximum length restriction for the resulting format trigger.
- If multiple exceptions exist, they are controlled by an implicit OR relation, which means that as soon as one of the exceptions has been applied (that is, satisfied), no other exceptions will be processed.

Example:
The following example shows two formatting exceptions for field f_ename. The first exception changes the text color to red if both of its conditions are met. The second exception changes the text color to blue if its condition is met.

```xml
<field name="f_ename" source="ename" label="Employee" textColor="green">
  <exception textColor="red">
    <condition source="deptno" operator="btw" operand1="1" operand2="20" relation="and"/>
    <condition source="sal" operator="gt" operand1="1000"/>
  </exception>
  <exception textColor="blue">
    <condition source="deptno" operator="eq" operand1="30"/>
  </exception>
</field>
```
7.5.8 <field>

**Description:** The `<field>` tag defines a field in the layout of the report definition and assigns attributes to it. The `<field>` tag must be nested within the `<layout>` tag. Most of the other layout tags require a `<field>` nested within them (for example, `<tabular>`, `<group>`, `<matrixCell>`). The `<field>` tag modifies existing fields in an .RDF file, if you use the same field name. Otherwise, it can be used to create an entirely new field in the report.

The `<field>` tag can also contain the `<labelAttribute>` and `<exception>` tags.

**Tip:** You can end the `<field>` tag with `/` or `</field>`. The latter is the method you must use if you are including an `<exception>` or `<labelAttribute>` inside the `<field>` tag. The example below illustrates both methods of ending the `<field>` tag.

```xml
<field name="f_deptno" label="Department" source="deptno"/>
<field name="f_mgr" label="Manager" source="mgr">
    <labelAttribute textColor="red" alignment="center"/>
</field>
```

For more information refer to:

Section 7.5.7, "<exception>"

Section 7.5.15, "<labelAttribute>"

**Syntax:**

```xml
<field
    name="field_name"
    source="source_column"
    [label="field_label"]
    [currency="currency_symbol"]
    [tsep="separator_character"]
    [formatTrigger="plsql_program_unit"]
    [font="font_name"]
    [fontSize="point_size"]
    [fontStyle="regular | italic | bold | boldItalic"]
    [fontEffect="regular | strikeout | underline | strikeoutUnderline"]
    [lineColor="color_name | noLine"]
    [fillColor="color_name | noFill"]
    [textColor="color_name"]
    [alignment="start | left | center | right | end"]
    [hyperlink="URL"]
    [linkdest="hyperlink_target"]
    [formatMask="mask"]
/> | >[other_tags]<field>
```
Attributes:
The following table describes the attributes of the `<field>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>required</td>
<td>Is the identifier for the field. If the name matches that of a field in an .RDF file to which the XML is being applied, the attributes specified will override those in the .RDF file.</td>
</tr>
<tr>
<td>source</td>
<td>required, for creating new fields, optional, for modifying existing fields</td>
<td>Is the source column from which the field gets its data. The source column must exist in the data model.</td>
</tr>
<tr>
<td>label</td>
<td>optional</td>
<td>Is the boilerplate text to be associated with the field. To control the formatting attributes of the label, you must use the <code>&lt;labelAttribute&gt;</code> tag. For more information refer to Section 7.5.15, &quot;&lt;labelAttribute&gt;&quot;. The label attribute only affects new fields, it will not change the label of an existing field in the .RDF file. To change the label of an existing field, you can use the <code>&lt;object&gt;</code> tag. For more information, refer to Section 7.5.22, &quot;&lt;object&gt;&quot;.</td>
</tr>
<tr>
<td>currency</td>
<td>optional</td>
<td>Is the currency symbol to be used with the field (for example, $). Note that you must still specify the formatMask attribute to indicate where you want the currency symbol placed.</td>
</tr>
<tr>
<td>tsep</td>
<td>optional</td>
<td>Is the separator character that you want to use when generating delimited output. The most commonly used delimiter is a tab, which can be read by spreadsheet programs such as Microsoft Excel.</td>
</tr>
<tr>
<td>formatTrigger</td>
<td>optional</td>
<td>Is the name of a PL/SQL program unit that is to be used as the format trigger for the field. Note that format triggers must be functions. For more information refer to the Report Builder online help system and look for format trigger in the index.</td>
</tr>
<tr>
<td>font</td>
<td>optional</td>
<td>Is the name of the font to be used for the field contents.</td>
</tr>
</tbody>
</table>
### Table 7–4  `<field>` Tag Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fontSize</td>
<td>optional</td>
<td>Is the size of the font to be used for the field contents.</td>
</tr>
<tr>
<td>fontStyle</td>
<td>optional</td>
<td>Is the style of the font to be used for the field contents:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ regular</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ italic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ boldItalic</td>
</tr>
<tr>
<td>fontEffect</td>
<td>optional</td>
<td>Is the effect of the font to be used for the field contents:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ regular</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ strikeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ underline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ strikeoutUnderline</td>
</tr>
<tr>
<td>lineColor</td>
<td>optional</td>
<td>Is the name of the color to be used for the border of the field. If noLine is specified, the field’s border will be transparent (that is, invisible).</td>
</tr>
<tr>
<td>fillColor</td>
<td>optional</td>
<td>Is the name of the color to be used as the background for the field. If noFill is specified, the background will be transparent.</td>
</tr>
<tr>
<td>textColor</td>
<td>optional</td>
<td>Is the name of the color to be used for the field contents.</td>
</tr>
<tr>
<td>alignment</td>
<td>optional</td>
<td>Is how the text should be justified within the field:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ center</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ right</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ end</td>
</tr>
<tr>
<td>hyperlink</td>
<td>optional</td>
<td>Is a URL to be associated with the field contents when HTML or PDF output is generated. This attribute is ignored for other types of output such as PostScript or ASCII.</td>
</tr>
</tbody>
</table>
The following example shows a section in the layout of a report definition that defines fields within two break groups for a matrix report:

```xml
<group>
  <field name="f_quarter" source="quarter" label="Quarter:" font="Arial" fontSize="8" formatTrigger="F_quarterFormatTrigger">
    <labelAttribute font="Arial" fontSize="8" fontStyle="bold" textColor="black"/>
  </field>
  <field name="f_SumTOTAL_SALESPerQUARTER" source="SumTOTAL_SALESPerQUARTER" label="Qtrly:  Sales: " font="Arial" fontSize="8" fontStyle="bold" formatMask="LNNGNNNGNNNGNN0D00">  
    <labelAttribute font="Arial" fontSize="8" fontStyle="bold" textColor="black"/>
  </field>
  <field name="f_SumTOTAL_COSTPerQUARTER" source="SumTOTAL_COSTPerQUARTER" label="Costs: " font="Arial" fontSize="8" fontStyle="bold" formatMask="LNNGNNNGNNNGNN0D00">  
    <labelAttribute font="Arial" fontSize="8" fontStyle="bold" textColor="black"/>
  </field>
  <field name="f_SumTOTAL_PROFITPerQUARTER" source="SumTOTAL_PROFITPerQUARTER" label="Profits: " font="Arial" fontSize="8" fontStyle="bold" formatMask="LNNGNNNGNNNGNN0D00">  
    <labelAttribute font="Arial" fontSize="8" fontStyle="bold" textColor="black"/>
  </field>
</group>
```
The following example shows a section in the layout of a report definition that defines a field within a break group for a group left report. Notice that the formatTrigger attribute points to a function that would be defined within the <programUnits> tag.

```
<group>
  <field name="f_quarter1" source="quarter1" label="Quarter"
  font="Arial" fontSize="8"
  formatTrigger="F_quarter1FormatTrigger">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
  </field>
</group>
```

### 7.5.9 `<formLike>`

**Description:** The `<formLike>` tag delimits a form style within a section of the report’s layout. If you use the `<formLike>` tag, you must also nest `<field>` tags to list the fields you want to include in the form layout.

For more information on the `<field>` tag, refer to Section 7.5.8, "<field>".

**Syntax:**

```
<formLike>
  <field>
    ...
  </field>
</formLike>
```
Example: The following example shows a segment of an XML report definition that defines a section with a form layout inside of it.

```xml
<section name="main">
  <formLike name="M_empform" template="corp2.tdf">
    <labelAttribute textColor="green" alignment="center"/>
    <field name="f_deptno" source="deptno" label="Department"/>
    <field name="f_mgr" source="mgr" label="Manager">
      <labelAttribute textColor="red" alignment="center"/>
    </field>
    <field name="f_job" label="Job" source="job"/>
  </formLike>
</section>
```

7.5.10 `<formula>`

**Description:** The `<formula>` tag defines a formula column in the data model of the report definition. A formula column uses a PL/SQL function to perform an operation, typically a complex calculation of some kind. If you are performing a common calculation (for example, sum, percent of total, or standard deviation), you can use the `<summary>` tag, which requires no PL/SQL.

For more information refer to:

Section 7.5.29, "<summary>"

**Syntax:**

```xml
<formula
  name="column_name"
  source="plsql_function_name"
  dataType="number | character | date"
  width="number"
/>
```

**Attributes:** The following table describes the attributes of `<formula>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>required</td>
<td>Is the name of the formula column.</td>
</tr>
<tr>
<td>source</td>
<td>required</td>
<td>Is the name of a PL/SQL function defined within the <code>&lt;programUnits&gt;</code> tag that performs the desired operation for the formula.</td>
</tr>
</tbody>
</table>
Example:

The following example shows a segment of an XML report definition that defines a data model with a formula column in it. Note that the defaulting algorithm will place the column in the appropriate group based on where we place its associated fields in the <layout> section.

```xml
<data>
  <dataSource name="Q_1">
    <select>
      SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER,
            VIDEO_CATEGORY_BY_QTR.SALES_REGION,
            VIDEO_CATEGORY_BY_QTR.STATE, VIDEO_CATEGORY_BY_QTR.CITY,
            VIDEO_CATEGORY_BY_QTR.PRODUCT_CATEGORY,
            VIDEO_CATEGORY_BY_QTR.TOTAL_SALES,
            VIDEO_CATEGORY_BY_QTR.TOTAL_COST, VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT
      FROM SCOTT.VIDEO_CATEGORY_BY_QTR
      WHERE VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
    </select>
  </dataSource>
  <dataSource name="Q_2">
    <select>
      SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER, VIDEO_CATEGORY_BY_QTR.CITY,
            VIDEO_CATEGORY_BY_QTR.PRODUCT_CATEGORY,
            VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT,
            VIDEO_CATEGORY_BY_QTR.TOTAL_SALES,
            VIDEO_CATEGORY_BY_QTR.TOTAL_COST
      FROM SCOTT.VIDEO_CATEGORY_BY_QTR
      WHERE VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
    </select>
  </dataSource>
</data>
```

### Table 7–5 &lt;formula&gt; Tag Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataType</td>
<td>optional</td>
<td>Is the type of data that will be generated by the formula. For example, if the formula performs a mathematical operation, the result will be a number. The possible values for dataType are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- character</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- date</td>
</tr>
<tr>
<td>width</td>
<td>optional</td>
<td>Is the number of characters wide of the result of the formula.</td>
</tr>
</tbody>
</table>
<formula name="Profit_Margin" source="FormulaProfitMargin"
datatype="number" width="9"/>
</data>
<programUnits>
  <function name="FormulaProfitMargin">
    <![CDATA[
      FUNCTION FormulaProfitMargin RETURN number IS
      BEGIN
        return ((:TOTAL_PROFIT1 / (:TOTAL_SALES1 - (0.07 * :TOTAL_SALES1))) * 100);
        END;
      ]]>
    </function>
</programUnits>

7.5.11 <function>

The <function> tag defines a PL/SQL function that you want to add to the report definition. The <function> tag must be nested within a <programUnits> tag. To reference a function, you use the formatTrigger attribute of the <field> tag.

For more information refer to:
Section 7.5.23, "<programUnits>"
Section 7.5.8, "<field>"

Syntax:

<function
  name="function_name"
>
  PLSQL_function
</function>

Attributes: The following table describes the attributes of the <function> tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>required</td>
<td>Is the identifier for the function. This is the name that should be used when referencing the function (for example, from the formatTrigger attribute of the &lt;field&gt; tag).</td>
</tr>
</tbody>
</table>
Example: The following example shows a segment of an XML report definition that defines some PL/SQL functions. Notice that the functions are referenced from fields in the layout through the formatTrigger attribute.

```xml
<layout>
  <section name="header">
    <field name="F_ssn1"
      source="ssn1"
      formatTrigger="F_ssn1FormatTrigger"/>
  </section>
  <section name="main">
    <field name="F_ssn"
      source="ssn"
      formatTrigger="F_ssnFormatTrigger"/>
  </section>
</layout>
<programUnits>
  <function name="F_ssn1FormatTrigger">
    <![CDATA[
      function F_ssn1FormatTrigger return boolean is
        begin
          SRW.SET_HYPERLINK('#EMP_DETAILS_&<'||LTRIM(TO_CHAR(:SSN))||'>');
          return (TRUE);
        end;
    ]]>
  </function>
  <function name="F_ssnFormatTrigger">
    <![CDATA[
      function F_ssnFormatTrigger return boolean is
        begin
          SRW.SET_LINKTAG('EMP_DETAILS_&<'||LTRIM(TO_CHAR(:SSN))||'>');
          return (TRUE);
        end;
    ]]>
  </function>
</programUnits>
```
7.5.12 <group>

Description: The <group> tag delimits the master group in a master-detail style layout. The <group> tag can only be nested within a <groupLeft>, <groupAbove>, or <matrix> tag. You must nest <field> tags within the <group> tag to list the fields you want to include in the master group.

For more information refer to:
Section 7.5.8, "<field>"
Section 7.5.13, "<groupAbove>"
Section 7.5.14, "<groupLeft>"
Section 7.5.18, "<matrix>"

Syntax: 
<group>
  master_group_content
</group>

Example: The following example shows a section in the layout of a report definition that defines fields within two break groups for a matrix report.

```xml
<group>
  <field name="f_quarter" source="quarter" label="Quarter:
  font="Arial" fontSize="8"
  formatTrigger="F_quarterFormatTrigger">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="black"/>
  </field>
  <field name="f_SumTOTAL_SALESPerQUARTER" source="SumTOTAL_SALESPerQUARTER"
    label="Qtrly:  Sales: " font="Arial" fontSize="8" fontStyle="bold"
    formatMask="LNNNGNNNGNNNGNN0D00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="black"/>
  </field>
  <field name="f_SumTOTAL_COSTPerQUARTER" source="SumTOTAL_COSTPerQUARTER"
    label="Costs: " font="Arial" fontSize="8" fontStyle="bold"
    formatMask="LNNNGNNNGNNGNOD00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="black"/>
  </field>
  <field name="f_SumTOTAL_PROFITPerQUARTER" source="SumTOTAL_PROFITPerQUARTER"
    label="Profits: " font="Arial" fontSize="8" fontStyle="bold"
    formatMask="LNNNGNNNGNNGNOD00">
</group>
```
7.5.13 <groupAbove>

**Description:** The `<groupAbove>` tag delimits a master-detail style within a section of the report’s layout. The master records will be placed above the detail records. If you use the `<groupAbove>` tag, you must also nest a `<group>` tag to identify the master group as well as `<field>` tags to list the fields you want to include in the group above layout.

For more information refer to:
- Section 7.5.8, "<field>"
- Section 7.5.12, "<group>"

**Syntax:**
```
<groupAbove
    name="style_name"
>
    <group>
        master_group_content
    </group>
    detail_group_content
</groupAbove>
```

**Example:** The following example shows a segment of an XML report definition that defines a section with a group above layout inside of it.
```
<section name="main">
    <groupAbove name="m_emp">
        <labelAttribute font="Arial" fontSize="10" fontStyle="bold" textColor="black"/>
        <field name="f_deptno" source="deptno" label="Department "
            font="Arial" fontSize="10"/>
        <field name="f_sumsal" label="Total Salary" source="sumsal" textColor="red" font="Arial" fontSize="10"
```

7.5.14 <groupLeft>

**Description:** The `<groupLeft>` tag delimits a master-detail style within a section of the report’s layout. The master records will be placed to the left of the detail records. If you use the `<groupLeft>` tag, you must also nest a `<group>` tag to identify the master group as well as `<field>` tags to list the fields you want to include in the group left layout.

For more information refer to:
Section 7.5.8, "<field>"
Section 7.5.12, "<group>"

**Syntax:**
```
<groupLeft
    name="style_name"
>
    <group>
        master_group_content
    </group>
    detail_group_content
</groupLeft>
```

**Example:** The following example shows a segment of an XML report definition that defines a section with a group left layout inside of it.

```
<section name="main">
    <groupLeft name="m_emp">
        <labelAttribute font="Arial" fontSize="10" fontStyle="bold"
            textColor="red"/>
        </field>
    </group>
    <field name="f_deptno" source="deptno" label="Department "
        font="Arial" fontSize="10"/>
    <field name="f_sumsal" label="Total Salary" source="sumsal"
        textColor="red" font="Arial" fontSize="10"
        fontStyle="bold">
7.5.15 <labelAttribute>

Description: The <labelAttribute> tag defines the formatting attributes for field labels. The <labelAttribute> tag can be nested within a <field> tag or within a layout style tag (for example, <tabular> or <matrix>). If <labelAttribute> is nested inside a <field> tag, it applies only to the labels for that field.

The <labelAttribute> tag only affects new fields, it will not change the label of an existing field in the .RDF file. Note that to change the text of an existing label, you should use the textSegment attribute of the <property> tag.

For more information refer to:
Section 7.5.8, "<field>"
Section 7.5.25, "<property>"

Syntax:
<labelAttribute
[font="font_name"]
[fontSize="point_size"]
[fontStyle="regular | italic | bold | boldItalic"]
[fontEffect="regular | strikeout | underline | strikeoutUnderline"]
[lineColor="color_name | noLine"]
[fillColor="color_name | noFill"]
[textColor="color_name"]
[alignment="start | left | center | right | end"]
>
</labelAttribute>
Attributes: The following table describes the attributes of the `<labelAttribute>` tag.

### Table 7–7 `<labelAttribute>` Tag Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>font</td>
<td>optional</td>
<td>Is the name of the font to be used for the field label.</td>
</tr>
<tr>
<td>fontSize</td>
<td>optional</td>
<td>Is the size of the font to be used for the field label.</td>
</tr>
<tr>
<td>fontStyle</td>
<td>optional</td>
<td>Is the style of the font to be used for the field label:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ regular</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ italic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ bold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ boldItalic</td>
</tr>
<tr>
<td>fontEffect</td>
<td>optional</td>
<td>Is the effect of the font to be used for the field contents:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ regular</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ strikeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ underline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ strikeoutUnderline</td>
</tr>
<tr>
<td>lineColor</td>
<td>optional</td>
<td>Is the name of the color to be used for the border of the field. If noLine is specified, the field's border will be transparent (that is, invisible).</td>
</tr>
<tr>
<td>fillColor</td>
<td>optional</td>
<td>Is the name of the color to be used as the background for the field. If noFill is specified, the background will be transparent.</td>
</tr>
<tr>
<td>textColor</td>
<td>optional</td>
<td>Is the name of the color to be used for the field contents.</td>
</tr>
<tr>
<td>alignment</td>
<td>optional</td>
<td>Is how the text should be justified within the field:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ center</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ right</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ end</td>
</tr>
</tbody>
</table>
Example: The following example shows a segment of an XML report definition that defines a section with a group left layout inside of it. Notice that the first <labelAttribute> tag would apply to all of the fields in the layout except for f_sumsal, which has its own embedded <labelAttribute> tag.

```xml
<section name="main">
    <groupLeft name="m_emp">
        <labelAttribute font="Arial" fontSize="10" fontStyle="bold"/>
        <group>
            <field name="f_deptno" source="deptno" label="Department " font="Arial" fontSize="10"/>
            <field name="f_sumsal" label="Total Salary" source="sumsal" textColor="red" font="Arial" fontSize="10" fontStyle="bold"> <labelAttribute font="Arial" fontSize="10" fontStyle="bold" textColor="red"/></field>
        </group>
        <field name="f_ename" source="ename" label="Name" font="Arial" fontSize="10"/>
        <field name="f_sal" source="sal" label="Salary" font="Arial" fontSize="10"/>
    </groupLeft>
</section>
```

7.5.16 <layout>

Description: The <layout> tag delimits the beginning and ending of the layout of the report definition.

Syntax: `<layout>
         content_of_layout
     </layout>`

Examples: The following example shows the layout segment of an XML report definition. Note that this is not a complete layout model and would have to be applied as a customization to an .RDF file.

```xml
<layout>
    <section name="main">
        <field name="f_trade_date" source="trade_date" formatMask="MM/DD/RR"/>
        <field name="F_Mincurrent_pricePersymbol" source="Mincurrent_pricePersymbol" lineColor="black"/>
    </section>
</layout>
```
The following example shows another layout segment of an XML report definition. Note that this is a complete layout and, assuming the appropriate data model was in place, it could stand by itself, without being applied to an .RDF file.

```xml
<layout>
  <section name="main">
    <matrix name="M_video_sales" template="corp10.tdf">
      <group>
        <field name="f_quarter" source="quarter" label="Quarter:
font="Arial" fontSize="8"
formatTrigger="F_quarterFormatTrigger">
          <labelAttribute font="Arial" fontSize="8"
fontStyle="bold" textColor="black"/>
        </field>
        <field name="f_SumTOTAL_SALESPerQUARTER" source="SumTOTAL_SALESPerQUARTER"
label="Qtrly:  Sales: " font="Arial" fontSize="8"
fontStyle="bold"
formatMask="LNNNGNNNNNNNNNNN0D00">
          <labelAttribute font="Arial" fontSize="8"
fontStyle="bold" textColor="black"/>
        </field>
        <field name="f_SumTOTAL_COSTPerQUARTER" source="SumTOTAL_COSTPerQUARTER"
label="Costs: " font="Arial" fontSize="8" fontStyle="bold"
formatMask="LNNNGNNNNNNNNNNN0D00">
          <labelAttribute font="Arial" fontSize="8"
fontStyle="bold" textColor="black"/>
        </field>
        <field name="f_SumTOTAL_PROFITPerQUARTER" source="SumTOTAL_PROFITPerQUARTER"
label="Profits: " font="Arial" fontSize="8" fontStyle="bold"
formatMask="LNNNGNNNNNNNNNNN0D00">
          <labelAttribute font="Arial" fontSize="8"
fontStyle="bold" textColor="black"/>
        </field>
      </group>
    </matrix>
  </section>
</layout>
```
<group>
  <field name="f_state" source="state" label="State:">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="black"/>
  </field>
</group>

<matrixCol name="g_city">
  <field name="f_city" source="city" label="City: "
    font="Arial" fontSize="8" textColor="yellow"
    formatTrigger="F_cityFormatTrigger"/>
  <field name="f_SumTOTAL_SALESPerCITY" source="SumTOTAL_SALESPerCITY"
    label="Sales: " font="Arial" fontSize="8" fontStyle="bold"
    textColor="yellow" formatMask="LNNNNGNNNGNNNGNN0D00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
  </field>
  <field name="f_SumTOTAL_COSTPerCITY" source="SumTOTAL_COSTPerCITY"
    label="Sales: " font="Arial" fontSize="8" fontStyle="bold"
    textColor="yellow" formatMask="LNNNNGNNNGNNNGNN0D00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
  </field>
  <field name="f_SumTOTAL_PROFITPerCITY" source="SumTOTAL_PROFITPerCITY"
    label="Sales: " font="Arial" fontSize="8" fontStyle="bold"
    textColor="yellow" formatMask="LNNNNGNNNGNNNGNN0D00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
  </field>
</matrixCol>

<matrixRow name="g_product_category">
  <field name="f_product_category" source="product_category"
    label="Product Category" font="Arial" fontSize="8"/>
</matrixRow>

<matrixCell name="g_total_sales">
  <field name="f_total_sales" source="total_sales" label="Total Sales"
    font="Arial" fontSize="8" lineColor="noLine"
    formatMask="LNNNNGNNNGNNNGNN0D00"/>
  <field name="f_total_cost" source="total_cost" label="Total Cost"
    font="Arial" fontSize="8" lineColor="noLine"
    formatMask="LNNNNGNNNGNNNGNN0D00"/>
  <field name="f_total_profit" source="total_profit" label="Total Profit"
    font="Arial" fontSize="8" lineColor="noLine"
    formatMask="LNNNNGNNNGNNNGNN0D00"/>
</matrixCell>
7.5.17 <link>

Description: The <link> tag defines a link between data sources in the data model. <link> must be nested within the <data> tag. Data sources are linked by columns. Hence each column link requires parent and child column attributes and a condition attribute that relates the columns. In order to join two tables or views, the foreign key columns must have a column alias in the SELECT statements. (These aliases are used to reference the parent and child column in the column link specification.)

Syntax:  
```xml
<link
  parentGroup="parent_group_name"
  parentColumn="parent_column_name"
  childQuery="child_query_name"
  childColumn="child_column_name"
  condition="eq | lt | lteq | neq | gt | gteq | like | notLike"
  sqlClause="startWith | having | where"
  name="link_name"
>
</link>
```

Attributes: The following table describes the attributes of the <link> tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentGroup</td>
<td>required for group links</td>
<td>Is the name of the parent group that you want to relate to the child query.</td>
</tr>
<tr>
<td></td>
<td>optional for column links</td>
<td></td>
</tr>
<tr>
<td>parentColumn</td>
<td>required for column links</td>
<td>Is the name of a column in the parent query that relates to a column in the child query (that is, child column).</td>
</tr>
<tr>
<td></td>
<td>ignored for group links</td>
<td></td>
</tr>
<tr>
<td>childQuery</td>
<td>required for group links</td>
<td>Is the name of the child query that relates to the parent group.</td>
</tr>
<tr>
<td></td>
<td>optional for column links</td>
<td></td>
</tr>
</tbody>
</table>
Table 7–8  <link> Tag Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>childColumn</td>
<td>required for column links ignored for group links</td>
<td>Is the name of a column in the child query that relates to a column in the parent query (that is, parent column).</td>
</tr>
<tr>
<td>condition</td>
<td>required</td>
<td>Is a SQL operator that defines the relationship between parent column and child column. Condition can have the following values: eq (equal to) lt (less than) lteq (less than or equal to) neq (not equal to) gt (greater than) gteq (greater than or equal to) Like (means that the condition is true when the value in one column matches the pattern in the other column. The pattern can contain % and _ as wildcard characters.) notLike (means that the condition is true when the value in one column does not match the pattern in the other column. The pattern can contain % and _ as wildcard characters.)</td>
</tr>
<tr>
<td>sqlClause</td>
<td>required</td>
<td>Is the type of SQL clause that relates the parent group to the child query. The default is a WHERE clause.</td>
</tr>
</tbody>
</table>

Example: The following example shows the data model segment of a report definition with a link between two queries.

```xml
<data>
  <dataSource name="Q_dept">
    <select>
      select deptno deptno_dept from dept
    </select>
  </dataSource>
  <dataSource name="Q_emp">
    <select>
      select deptno deptno_emp, ename, empno, sal from emp
    </select>
  </dataSource>
</data>
```
7.5.18 <matrix>

**Description:** The `<matrix>` tag delimits a matrix style within a section of the report’s layout. If you use the `<matrix>` tag, you must also nest `<matrixRow>`, `<matrixCol>`, and `<matrixCell>` tags to identify the parts of the matrix as well as `<field>` tags to list the fields you want to include in the matrix layout.

Tip: A `<group>` tag can also be used in conjunction with `<matrix>` tags to create a matrix with group style.

For more information refer to:

- Section 7.5.8, "<field>"
- Section 7.5.12, "<group>"
- Section 7.5.20, "<matrixCol>"
- Section 7.5.21, "<matrixRow>"
- Section 7.5.19, "<matrixCell>"

**Syntax:**

```xml
<matrix
    name="style_name"
>
    [<group
        master_group_content
    </group>]
    <matrixCol
        matrix_column_content
    </matrixCol>
    <matrixRow
        matrix_row_content
    </matrixRow>
    <matrixCell
        matrix_cell_content
    </matrixCell>
</matrix>
```
Example: The following example shows a segment of an XML report definition that defines a matrix with group layout.

```xml
<matrix name="M_video_sales" template="corp10.tdf">
  <group>
    <field name="f_quarter" source="quarter" label="Quarter:
      font="Arial" fontSize="8"
      formatTrigger="F_quarterFormatTrigger">
      <labelAttribute font="Arial" fontSize="8"
        fontStyle="bold" textColor="black"/>
    </field>
    <field name="f_SumTOTAL_SALESPerQUARTER" source="SumTOTAL_SALESPerQUARTER"
      label="Qtrly:  Sales: " font="Arial" fontSize="8"
      fontStyle="bold"
      formatMask="LNNNGNNGGNNN0D00">
      <labelAttribute font="Arial" fontSize="8"
        fontStyle="bold" textColor="black"/>
    </field>
    <field name="f_SumTOTAL_COSTPerQUARTER" source="SumTOTAL_COSTPerQUARTER"
      label="Costs: " font="Arial" fontSize="8"
      fontStyle="bold"
      formatMask="LNNNGNNGGNNN0D00">
      <labelAttribute font="Arial" fontSize="8"
        fontStyle="bold" textColor="black"/>
    </field>
    <field name="f_SumTOTAL_PROFITPerQUARTER" source="SumTOTAL_PROFITPerQUARTER"
      label="Profits: " font="Arial" fontSize="8"
      fontStyle="bold"
      formatMask="LNNNGNNGGNNN0D00">
      <labelAttribute font="Arial" fontSize="8"
        fontStyle="bold" textColor="black"/>
    </field>
  </group>
  <matrixCol name="g_city">
    <field name="f_city" source="city" label="City: 
      font="Arial" fontSize="8" textColor="yellow"
      formatTrigger="F_cityFormatTrigger"/>
    <field name="f_SumTOTAL_SALESPerCITY" source="SumTOTAL_SALESPerCITY"
XML Tag Reference

```
<row name="g_product_category">
  <field name="f_product_category" source="product_category"
    label="Product Category" font="Arial" fontSize="8" />
</row>

<row name="g_total_sales">
  <field name="f_total_sales" source="total_sales"
    label="Total Sales" font="Arial" fontSize="8" lineColor="noLine"
    formatMask="LNNNGNNNGNNNGNN0D00" />
  <field name="f_total_cost" source="total_cost"
    label="Total Cost" font="Arial" fontSize="8" lineColor="noLine"
    formatMask="LNNNGNNNGNNNGNN0D00" />
  <field name="f_total_profit" source="total_profit"
    label="Total Profit" font="Arial" fontSize="8" lineColor="noLine"
    formatMask="LNNNGNNNGNNNGNN0D00" />
</row>
```
7.5.19 <matrixCell>

**Description:** The `<matrixCell>` tag delimits the cells in a matrix style layout. The `<matrixCell>` tag can only be nested within a `<matrix>` tag. You must nest `<field>` tags within the `<matrixCell>` tag to list the fields you want to include as matrix cells.

For more information refer to:
Section 7.5.8, "<field>"
Section 7.5.18, "<matrix>"

**Syntax:**
```
<matrixCell>
    master_group_content
</matrixCell>
```

**Example:** The following example shows a segment of an XML report definition that defines a matrix cell.
```
<matrixCell name="g_total_sales">
    <field name="f_total_sales" source="total_sales" label="Total Sales"
        font="Arial" fontSize="8" lineColor="noLine"
        formatMask="LNNNNNNNNNNNN0D00"/>
    <field name="f_total_cost" source="total_cost" label="Total Cost"
        font="Arial" fontSize="8" lineColor="noLine"
        formatMask="LNNNNNNNNNNNN0D00"/>
    <field name="f_total_profit" source="total_profit" label="Total Profit"
        font="Arial" fontSize="8" lineColor="noLine"
        formatMask="LNNNNNNNNNNNN0D00"/>
</matrixCell>
```

7.5.20 <matrixCol>

**Description:** The `<matrixCol>` tag delimits the column fields in a matrix style layout. The `<matrixCol>` tag can only be nested within a `<matrix>` tag. You must nest `<field>` tags within the `<matrixCol>` tag to list the fields you want to include as matrix columns.

For more information refer to:
Section 7.5.8, "<field>"
Section 7.5.18, "<matrix>"

**Syntax:**
```
<matrixCol>
    master_group_content
</matrixCol>
```
Example: The following example shows a segment of an XML report definition that defines the column dimension of a matrix layout.

```xml
<matrixCol name="g_city">
  <field name="f_city" source="city" label="City: "
    font="Arial" fontSize="8" textColor="yellow"
    formatTrigger="F_cityFormatTrigger"/>
  <field name="f_SumTOTAL_SALESPerCITY" source="SumTOTAL_SALESPerCITY"
    label="Sales: " font="Arial" fontSize="8" fontStyle="bold"
    textColor="yellow" formatMask="LNNNGNNNNNNNNGNDD00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
  </field>
  <field name="f_SumTOTAL_COSTPerCITY" source="SumTOTAL_COSTPerCITY"
    label="Sales: " font="Arial" fontSize="8" fontStyle="bold"
    textColor="yellow" formatMask="LNNNGNNNNNNNNGNDD00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
  </field>
  <field name="f_SumTOTAL_PROFITPerCITY" source="SumTOTAL_PROFITPerCITY"
    label="Sales: " font="Arial" fontSize="8" fontStyle="bold"
    textColor="yellow" formatMask="LNNNGNNNNNNNNGNDD00">
    <labelAttribute font="Arial" fontSize="8"
      fontStyle="bold" textColor="yellow"/>
  </field>
</matrixCol>
```

7.5.21 <matrixRow>

Description: The <matrixRow> tag delimits the row fields in a matrix style layout. The <matrixRow> tag can only be nested within a <matrix> tag. You must nest <field> tags within the <matrixRow> tag to list the fields you want to include as matrix rows.

For more information refer to:
Section 7.5.8, "<field>"
Section 7.5.18, "<matrix>"

Syntax: <matrixRow>
  master_group_content
</matrixRow>
The following example shows a segment of an XML report definition that defines the row dimension of a matrix layout.

```xml
<matrixRow name="g_product_category">
    <field name="f_product_category" source="product_category"
          label="Product Category" font="Arial" fontSize="8"/>
</matrixRow>
```

### 7.5.22 `<object>`

**Description:** The `<object>` tag identifies an object in the report whose properties you want to change. The `<object>` tag typically has `<properties>` and `<property>` tags nested within it.

**Syntax:**

```xml
<object
    name="object_name"
    type="REP_REPORT | REP_GROUP | REP_COL_MAP | REP_GRAPHIC_TEXT"
>
    property_definitions
</object>
```

**Attributes:** The following table describes the attributes of the `<object>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>required</td>
<td>Is the identifier for the object to which you want to apply the properties.</td>
</tr>
<tr>
<td>type</td>
<td>required</td>
<td>Is the kind of object to which you want to apply the properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- REP_REPORT is the report itself.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- REP_GROUP is a group in the data model of the report.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- REP_COL_MAP is a column in the data model of the report.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- REP_GRAPHIC_TEXT is a boilerplate object in the layout of the report.</td>
</tr>
</tbody>
</table>
Example: The following example shows a segment of an XML report definition that defines some object properties.

```xml
<customize>
  <object name="videosales" type="REP_REPORT">
    <properties>
      <property name="beforeReportType">File</property>
      <property name="beforeReportValue">
        d:\xml_reps\header_example.html
      </property>
      <property name="afterReportType">Text</property>
      <property name="afterReportValue">
        <![CDATA[
          <center>
            <font face="Arial,Helvetica"><font size=-1><font color="#000000">
              Send questions to <a href="mailto:your_email_id">YourNameHere</a>.
              <br>
            </font></font></center>
          ]]>}
        </property>
    </properties>
  </object>
</customize>
```

The following example shows a segment of an XML report definition that changes some boilerplate text. This is useful for changing labels for existing fields.

```xml
<customize>
  <object name="B_high_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">High</property>
    </properties>
  </object>
  <object name="B_low_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">Low</property>
    </properties>
  </object>
</customize>
```
**7.5.23 <programUnits>**

**Description:** The <programUnits> tag delimits any PL/SQL that you want to add to the report definition. The <programUnits> tag typically has <function> tags nested within it. For more information refer to: Section 7.5.11, "<function>"

**Syntax:**

```xml
<programUnits>
  program_unit_definitions
</programUnits>
```

**Example:** The following example shows a segment of an XML report definition that defines some PL/SQL. Notice that the <programUnits> tag is outside of the <layout> tag and that the functions are referenced from fields in the layout through the formatTrigger attribute.

```xml
<layout>
  <section name="header">
    <field name="F_ssn1"
      source="ssn1"
      formatTrigger="F_ssn1FormatTrigger"/>
  </section>
  <section name="main">
    <field name="F_ssn"
      source="ssn"
      formatTrigger="F_ssnFormatTrigger"/>
  </section>
</layout>
<programUnits>
  <function name="F_ssn1FormatTrigger">
    <![CDATA[
      function F_ssn1FormatTrigger return boolean is
      begin
        SRW.SET_HYPERLINK('#EMP_DETAILS_&< ' || LTRIM(TO_CHAR(:SSN)) || '>' );
        return (TRUE);
      end;
    ]]>
  </function>
  <function name="F_ssnFormatTrigger">
    <![CDATA[
      function F_ssnFormatTrigger return boolean is
      begin
        SRW.SET_LINKTAG('EMP_DETAILS_&< ' || LTRIM(TO_CHAR(:SSN)) || '>' );
        return (TRUE);
      end;
    ]]>
  </function>
</programUnits>
```
7.5.24 <properties>

**Description:** The `<properties>` tag delimits the properties of the object. The `<properties>` tag must be nested inside of the `<object>` tag and typically has `<property>` tags nested within it.

**Syntax:**
```
<properties>
  property_definitions
</properties>
```

**Example:** The following example shows a segment of an XML report definition that defines an object's properties.

```xml
<customize>
  <object name="videosales" type="REP_REPORT">
    <properties>
      <property name="beforeReportType">File</property>
      <property name="beforeReportValue">d:\xml_reps\header_example.html</property>
      <property name="afterReportType">Text</property>
      <property name="afterReportValue">
        <![CDATA[
          <center>
            <font face="Arial,Helvetica"><font size=-1><font color="#000000">
            Send questions to <a href="mailto:your_email_id">YourNameHere</a>.
          </font></font></center>
        ]]>}
      </property>
    </properties>
  </object>
</customize>
```
The following example shows a segment of an XML report definition that changes some boilerplate text. This is useful for changing labels for existing fields.

```xml
<customize>
  <object name="B_high_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">High</property>
    </properties>
  </object>
  <object name="B_low_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">Low</property>
    </properties>
  </object>
</customize>
```

### 7.5.25 `<property>`

**Description:** The `<property>` tag delimits a single property of the object. The `<property>` tag must be nested inside of the `<properties>` tag and typically has some text nested within it to define the value of the property.

**Syntax:**

```xml
<property
  name="xmlTag | xmlAttribute | xmlSuppress | prologType | prolog |
  beforeReportValue | beforeReportType | afterReportValue | afterReportType |
  beforePageValue | beforePageType | afterPageValue | afterPageType |
  beforeFormValue | beforeFormType | afterFormValue | afterFormType |
  pageNavigationControlValue | pageNavigationControlType | textSegment
>
  property_value
</property>
```

**Attributes:** The following table describes the attributes of the `<property>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>required</td>
<td>Is the name of the property that you want to specify. Note that the available properties vary depending upon the type of object. Refer to the Usage Notes for more information.</td>
</tr>
</tbody>
</table>
**Usage Notes:** The following table lists the properties that are available for each type of object.

<table>
<thead>
<tr>
<th>Table 7–11 Valid Properties for Object Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
</tr>
<tr>
<td>Report object (REP_REPORT)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Group object (REP_GROUP)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Column object (REP_COL_MAP)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Boilerplate object (REP_GRAPHIC_TEXT)</td>
</tr>
</tbody>
</table>
Example: The following example shows a segment of an XML report definition that defines an object’s properties.

```xml
<customize>
  <object name="videosales" type="REP_REPORT">
    <properties>
      <property name="beforeReportType">File</property>
      <property name="beforeReportValue">
        d:\xml_reps\header_example.html
      </property>
      <property name="afterReportType">Text</property>
      <property name="afterReportValue">
        <![CDATA[
          <center>
            <font face="Arial,Helvetica"><font size=-1><font color="#000000">
              Send questions to <a href="mailto:your_email_id">YourNameHere</a>.
            </font></font></center>
        ]]>
      </property>
    </properties>
  </object>
</customize>
```

The following example shows a customization section that changes the text in a boilerplate object. This is useful for changing labels for existing fields.

```xml
<customize>
  <object name="B_high_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">High</property>
    </properties>
  </object>
  <object name="B_low_365" type="REP_GRAPHIC_TEXT">
    <properties>
      <property name="textSegment">Low</property>
    </properties>
  </object>
</customize>
```
7.5.26 <report>

**Description:** The `<report>` tag delimits the beginning and ending of the report definition. You can append attributes that apply to the entire report to the `<report>` tag.

**Syntax:**
```
<report DTDVersion=1.0
    [name="report_name"]
    [title="report_title"]
    [author="author_name"]
>
    content_of_report
</report>
```

**Example:** This example shows an XML customization document designed to be applied to an .RDF file named cond.rdf. Note that this example does not touch the data model. It only changes the formatting of some of the fields in the layout.

```
<report name="cond" DTDVersion="1.0">
<!-- This report assumes that the file named header_example.html is located in d:\ORANT\TOOL2\DOC60\US\RRR60. If it is not located there, the report will not run properly. -->
<layout>
    <section name="main">
        <field name="f_trade_date"
            source="trade_date"
            formatMask="MM/DD/RR"/>
        <field name="F_Mincurrent_pricePersymbol"
            source="Mincurrent_pricePersymbol"
            lineColor="black"
            fillColor="r100g50b50"/>
        <field name="F_Maxcurrent_pricePersymbol"
            source="Maxcurrent_pricePersymbol"
            lineColor="black"
            fillColor="r100g50b50"/>
    </section>
</layout>
<customize>
    <object name="videosales" type="REP_REPORT">
        <properties>
            <property name="beforeReportType">File</property>
            <property name="beforeReportValue">
                d:\xml_reps\header_example.html
            </property>
        </properties>
    </object>
</customize>
```
The following table describes the attributes of the `<report>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>optional</td>
<td>Records the name of the report. If the name is not specified, the default will be UNTITLED. If you plan to apply the report definition to an .RDF file, this name should be the same as the filename without the .RDF extension.</td>
</tr>
<tr>
<td>dtdVer</td>
<td>required</td>
<td>Records the version of the Reports DTD used to generate this XML report definition. Since the DTD can change between versions, any new reports definition must include information about which version was used. This permits backward compatibility in future releases.</td>
</tr>
<tr>
<td>title</td>
<td>optional</td>
<td>Places the specified title at the beginning of the report. When applying the definition title at an .RDF file, this title will override the existing report title.</td>
</tr>
<tr>
<td>author</td>
<td>optional</td>
<td>Records the name of the author.</td>
</tr>
</tbody>
</table>
7.5.27 <section>

**Description:** The `<section>` tag delimits the beginning and ending of a section in the layout of the report definition. The `<section>` tag must be nested within the `<layout>` tag. A report may have up to three sections in its layout.

For each section, you may also define a layout style using the following tags:

- Section 7.5.30, "<tabular>"
- Section 7.5.18, "<matrix>"
- Section 7.5.9, "<formLike>"
- Section 7.5.13, "<groupAbove>"
- Section 7.5.14, "<groupLeft>"

**Syntax:**

```
<section
  name= "header | main | trailer"
  width="section_width"
  height="section_height"
>
  section_contents
</section>
```

**Attributes:**

The following table describes the attributes of the `<section>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>required</td>
<td>Is the section’s name: header, main, or trailer.</td>
</tr>
<tr>
<td>width</td>
<td>optional</td>
<td>Is the width of one physical page (including the margin) in the unit of measurement of the report (for example, 8.5 inches).</td>
</tr>
<tr>
<td>height</td>
<td>optional</td>
<td>Is the height of one physical page (including the margin) in the unit of measurement of the report (for example, 11 inches).</td>
</tr>
</tbody>
</table>
Example: The following is an example of a `<section>` definition:

```xml
<layout>
    <section name="header">
        <field name="F_ssn1"
               source="ssn"
               formatTrigger="F_ssn1FormatTrigger"/>
    </section>
    <section name="main">
        <field name="F_ssn"
               source="ssn"
               formatTrigger="F_ssnFormatTrigger"/>
    </section>
</layout>
```

### 7.5.28 `<select>`

**Description:** The `<select>` tag delimits the beginning and ending of a SELECT statement within the data model. `<select>` must be nested within the `<dataSource>` tag.

**Syntax:**

```xml
<select>
    content_of_SELECT
</select>
```

**Examples:** The following example shows the data source segment of an XML report definition.

```xml
<data>
    <dataSource name="q_category">
        <select>
            SELECT ic.category, 
                    SUM (h.sales), 
                    AVG (h.high_365), 
                    AVG (h.low_365), 
                    AVG (h.div), 
                    AVG (h.p_e)
            FROM stock_history h, indcat ic
            WHERE h.symbol=ic.symbol
            GROUP BY ic.category
        </select>
    </dataSource>
</data>
```
**Tip:** A user parameter is automatically generated for you if you include it as a bind reference in a SELECT statement. For example:

```xml
<select>
    select * from dept where deptno > :p_dept;
</select>
```

This SELECT statement would cause a user parameter named `p_dept` to be automatically generated. Therefore, you would not need to manually create it in the report definition.

The following example shows a segment of an XML report definition that uses the `<![CDATA[]]>` tag to protect a SQL statement that contains a greater than sign.

```xml
<select>
    <![CDATA[
        SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER,
            VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT
        FROM SCOTT.VIDEO_CATEGORY_BY_QTR
        WHERE (VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
            AND VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT>2000)
    ]]>}
</select>
```

### 7.5.29 `<summary>`

**Description:** The `<summary>` tag defines a summary column in the data model of the report definition. Summary columns are used to perform some mathematical function on the data values of another column. If you want to perform a function that is not one of the standard summary functions, you can use the `<formula>` tag to create a formula column that uses PL/SQL to perform more complex calculations.

For more information refer to:

Section 7.5.10, "<formula>"

**Syntax:**

```xml
<summary
    source="src_col_name"
    function="sum|average|minimum|maximum|count|first|last|pctTotal|stddeviation|variance"
    compute="group+names"
    reset="group_name"
    productOrder="group_name"
    nullval="value_if_null"
/>```

Attributes:
The following table describes the attributes of the `<summary>` tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>required</td>
<td>Is the name of the column whose values will be summarized.</td>
</tr>
<tr>
<td>function</td>
<td>optional</td>
<td>Is the mathematical operation to be applied to produce the summary values: average calculates the average of the column’s values within the reset group. count counts the number of records within the reset group. first prints the column’s first value fetched for the reset group. last prints the column’s last value fetched for the reset group. maximum calculates the column’s highest value within the reset group. minimum calculates the column’s lowest value within the reset group. pctTotal calculates the column’s percent of the total within the reset group. stddeviation calculates the column’s positive square root of the variance for the reset group. sum calculates the total of the column’s values within the reset group. variance sums the squares of each column value’s distance from the mean value of the reset group and divides the total by the number of values minus 1.</td>
</tr>
</tbody>
</table>
### Customizing Reports at Runtime

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute</td>
<td>optional</td>
<td>Is the group over which a % of Total summary column is computed. Compute is used only for columns with a function of % of Total. This value determines the total of which each source column value is a percentage. When you calculate a percentage, you divide a value by a total (for example, SMITH's salary/total department salaries). Compute defines the total for a percentage calculation. For matrix reports, Compute At can be multiple groups. You can also set this attribute to page or report if you want to compute percentages over the total values on each page or over the entire report.</td>
</tr>
<tr>
<td>reset</td>
<td>optional</td>
<td>Is the group at which the summary column value resets to zero (if Function is Count), null (if Function is not Count), or nullval (if the summary has one). Reset determines if the summary is a running summary or a periodic (for example, group-level) summary. You can also set this attribute to page or report if you want to compute percentages over the total values on each page or over the entire report.</td>
</tr>
<tr>
<td>productOrder</td>
<td>optional</td>
<td>Is the order in which groups are evaluated in the cross product for a summary. ProductOrder also defines the frequency of a summary, formula, or placeholder in a cross product group. That is, the summary, formula, or placeholder has one value for each combination of values of the groups in its productOrder. productOrder is used only for columns owned by cross-product groups. Because a cross product relates multiple groups, the groups in the cross product could be evaluated in any one of many different orders. Therefore, when creating a summary for a cross product, you must use productOrder to specify which group should be evaluated first, which second, and so on. You must also use productOrder to specify the frequency of a summary, formula, or placeholder within the cross product.</td>
</tr>
</tbody>
</table>
Table 7–14  <summary> Tag Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nullval</td>
<td>optional</td>
<td>Is a value to be substituted for any null values of the column. For example, if you enter X in this field, then an X will be displayed for null values fetched for the column. If left blank, no substitution will be done for null values.</td>
</tr>
</tbody>
</table>

Default Values: Typically, you should not need to specify anything for the optional attributes of the <summary> tag because their values are defaulted at runtime. The only time you should need to specify the optional values is when you want to override their defaults. The following tables describe the defaulting for each of the optional attributes for each layout style.

Table 7–15  Default Values for Summaries in Break Groups

<table>
<thead>
<tr>
<th>Optional Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>sum</td>
</tr>
<tr>
<td>compute</td>
<td>the parent group of the summary column’s group</td>
</tr>
<tr>
<td>reset</td>
<td>the parent group of the summary column’s group</td>
</tr>
</tbody>
</table>

Table 7–16  Default Values for Summaries in a Matrix Report

<table>
<thead>
<tr>
<th>Optional Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>sum</td>
</tr>
<tr>
<td>compute</td>
<td>the cross product group</td>
</tr>
<tr>
<td>productOrder</td>
<td>■ the group containing the summary (for dimension summaries)</td>
</tr>
<tr>
<td></td>
<td>■ a list of groups that define the matrix row (for cell summaries)</td>
</tr>
<tr>
<td>reset</td>
<td>the highest frequency group of the productOrder</td>
</tr>
</tbody>
</table>
Example:
The following is an example of some summaries for a data model that contains two queries. The first three summaries are for a tabular layout and the last six are for a matrix break report. Note that because only the name, source column, and function are specified, the defaulting algorithm will place the columns in the appropriate groups based on where we place their associated fields in the layout.

```xml
<data>
  <dataSource name="Q_1">
    <select>
      SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER,
      VIDEO_CATEGORY_BY_QTR.SALES_REGION,
      VIDEO_CATEGORY_BY_QTR.STATE, VIDEO_CATEGORY_BY_QTR.CITY,
      VIDEO_CATEGORY_BY_QTR.PRODUCT_CATEGORY,
      VIDEO_CATEGORY_BY_QTR.TOTAL_SALES,
      VIDEO_CATEGORY_BY_QTR.TOTAL_COST,
      VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT
      FROM SCOTT.VIDEO_CATEGORY_BY_QTR
      WHERE VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
    </select>
  </dataSource>
  <dataSource name="Q_2">
    <select>
      SELECT ALL VIDEO_CATEGORY_BY_QTR.QUARTER, VIDEO_CATEGORY_BY_QTR.CITY,
      VIDEO_CATEGORY_BY_QTR.PRODUCT_CATEGORY,
      VIDEO_CATEGORY_BY_QTR.TOTAL_PROFIT,
      VIDEO_CATEGORY_BY_QTR.TOTAL_SALES,
      VIDEO_CATEGORY_BY_QTR.TOTAL_COST
      FROM SCOTT.VIDEO_CATEGORY_BY_QTR
      WHERE VIDEO_CATEGORY_BY_QTR.SALES_REGION='West'
    </select>
  </dataSource>
  <summary name="SumTOTAL_SALESPerCITY1" source="total_sales1"/>
  <summary name="SumTOTAL_COSTPerCITY1" source="total_cost1"/>
  <summary name="SumTOTAL_PROFITPerCITY1" source="total_profit1"/>
  <summary name="SumTOTAL_SALESPerQUARTER" source="total_sales"/>
  <summary name="SumTOTAL_COSTPerQUARTER" source="total_cost"/>
  <summary name="SumTOTAL_PROFITPerQUARTER" source="total_profit"/>
  <summary name="SumTOTAL_SALESPerCITY" source="total_sales"/>
  <summary name="SumTOTAL_COSTPerCITY" source="total_cost"/>
  <summary name="SumTOTAL_PROFITPerCITY" source="total_profit"/>
  <formula name="Profit_Margin" source="FormulaProfitMargin"
    datatype="number"
    width="9"/>
</data>
```
7.5.30 <tabular>

Description: The <tabular> tag delimits a tabular style within a section of the report’s layout. If you use the <tabular> tag, you must also nest <field> tags to list the fields you want to include in the tabular layout.

For more information, refer to:
Section 7.5.8, "<field>"

Syntax: <tabular>
         <field>
         ...
         </field>
         </tabular>

Example: The following example shows a segment of an XML report definition that defines a section with a tabular layout inside of it.

```xml
<section name="header">
  <tabular name="M_summary" template="corp2.tdf">
    <labelAttribute font="Arial"
      fontSize="10"
      fontStyle="bold"
      textColor="white"/>
    <field name="F_ports"
      source="ports"
      label="Port IDs"
      font="Arial"
      fontSize="10"/>
    <field name="F_locations"
      source="locations"
      label="Port Names"
      font="Arial"
      fontSize="10"/>
  </tabular>
</section>
```
Appendix A, "RWCLI60 Command Line Arguments"
Appendix B, "Reports Server Configuration Parameters"
Appendix C, "Environment Variables"
Appendix D, "Database Connection Strings"
Appendix E, "Reports Web Cartridge"
Appendix F, "Troubleshooting"
RWCLI60 Command Line Arguments

This appendix contains descriptions of RWCLI60 command line arguments.

RWCLI60 parses and transfers the command line to the specified Reports Server (RWMTS60). It uses a command line very similar to RWRUN60.

**Syntax**

```
RWCLI60 [MODULE|REPORT=runfile USERID=userid 
[ [keyword=value|value1, value2, ... ] ] SERVER=tnsname
```

where `keyword=value` is a valid command line argument.

**Usage notes**

- All file names and paths specified in the client command line refer to files and directories on the server machine, except for command file.
- If the command line contains CMDFILE=, the command file will be read and appended to the original command line before being sent to the Reports Server. The runtime engine will not re-read the command file.

**MODULE|REPORT**

*Description*  
`MODULE | REPORT` is the name of the report to run. (REPORT is allowed for backward compatibility.)

*Syntax*

```
[MODULE|REPORT=]runfile
```
Values

Any valid runfile (that is, a file with an extension of .RDF, .REP, or .XML). If you do not enter a file extension, then Reports Runtime searches first for a file with extension .REP, then extension .RDF, then .XML, and then no extension. Reports Runtime will use its file path search order to find the file.

USERID

Description USERID is your ORACLE username or placeholder username (that is, $username) and password with an optional database name, Net8 communication protocol to access a remote database, or ODBC datasource name (if accessing a non-Oracle datasource). If the password is omitted, then a database logon form is provided.

If you want users to log on to the database, then omit the USERID command line argument from the report request. If you want users to log on every time they run report requests, then use the Web CGI command SHOWAUTH and AUTHTYPE=S in the report URL, or include the %D argument to the key mapping entry in the cgicmd.dat (Web CGI) file.

Values

The logon definition must be in one of the following forms and cannot exceed 512 bytes in length:

username[/password]
username[/password] [@database]
 [user[/password]]@ODBC:datasource[:database] or [user[/password]]@ODBC:*

<$username>[/password]
<$username>[/password] [@database]

See Appendix D, "Database Connection Strings" for a list of valid connection strings.
**PARAMFORM**
*Description* If PARAMFORM is specified, then it must be NO.

*Syntax*

```
[PARAMFORM=]NO
```

**CMDFILE**
*Description* CMDFILE is a file that contains arguments for the RWRUN60 command. This option enables you to run a report without having to specify a large number of arguments each time you invoke RWRUN60.

*Syntax*

```
[CMDFILE=]cmdfile
```

*Values*

Any valid command file.

*Restrictions*

- A command file may reference another command file.
- Command file syntax for RWRUN60 arguments is identical to that used on the command line.
- Values entered on the command line override values specified in command files. For example, suppose that you specify RWRUN60 from the command line with COPIES equal to 1 and CMDFILE equal to RUNONE (a command file). In RUNONE, COPIES is set to 2. Only one copy of the report would be generated in this case.
- The argument or arguments for this keyword may be operating system-specific.
TERM

Description TERM is the type of terminal on which you are using RWRUN60. TERM is useful for the Runtime Parameter Form and Runtime Previewer only. This keyword is only used in character mode.

Syntax

\[\text{TERM=}\text{termtype}\]

Values

Any valid terminal type.

Default

Installation dependent. (See your Reports Developer system administrator for a compatible definition.)

Usage note

The argument or arguments for this keyword may be case sensitive, depending on your operating system.

ARRAYSIZE

Description ARRAYSIZE is the size (in kilobytes) for use with ORACLE array processing. Generally, the larger the array size, the faster the report will run.

Syntax

\[\text{ARRAYSIZE=}\text{n}\]

Values

A number from 1 through 9,999. This means that Reports Runtime can use this number of kilobytes of memory per query in your report.

Default

The default array size is 10K. For details about the ORACLE array processing, see the Oracle8 Server Administrator’s Guide.
**DESTYPE**

**Description** DESTYPE is the type of device that will receive the report output.

**Syntax**

```
[DESTYPE=] {CACHE|LOCALFILE|FILE|PRINTER|SYSOUT|MAIL}
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACHE</td>
<td>Sends the output directly to the Reports Server’s cache. Note that DESTYPE=CACHE is not compatible with the DISTRIBUTE keyword. If the server encounters DISTRIBUTE on the command line, it will ignore the DESTYPE=CACHE command line argument.</td>
</tr>
<tr>
<td>LOCALFILE</td>
<td>Sends the output to a file on the client machine and forces a synchronous call, regardless of the BACKGROUND value.</td>
</tr>
<tr>
<td>FILE</td>
<td>Sends the output to the file on the server machine named in DESNAME.</td>
</tr>
<tr>
<td>PRINTER</td>
<td>Sends the output to the printer on the server machine named in DESNAME. You must have a printer that Reports Developer can recognize installed and running.</td>
</tr>
<tr>
<td>MAIL</td>
<td>Sends the output to the mail users specified in DESNAME. You can send mail to any mail system that is MAPI compliant or has the service provider driver installed. The report is sent as an attached file.</td>
</tr>
<tr>
<td>SYSOUT</td>
<td>Sends the output to the client machine’s default output device and forces a synchronous call.</td>
</tr>
</tbody>
</table>

**Default**

Taken from the Initial Value property of the DESTYPE parameter.

**Usage note**

Screen and Preview cannot be used for DESTYPE with RWCLI60.
DESNAME

Description DESNAME is the name of the file, printer, or e-mail ID (or distribution list) to which the report output will be sent. To send the report output via e-mail, specify the e-mail ID as you do in your e-mail application (any MAPI-compliant application on Windows or your native mail application on UNIX). You can specify multiple usernames by enclosing the names in parentheses and separating them by commas (for example, (name, name, . .name)).

Syntax

[DESNAME=]desname

Values

Any valid filename, printer name, or e-mail ID not to exceed 1K in length. For printer names, you can optionally specify a port. For example:

DESNAME=printer,LPT1:
DESNAME=printer,FILE:

Default

Taken from the Initial Value property of the DESNAME parameter. If DESTYPE=FILE and DESNAME is an empty string, then it defaults to reportname.lis at runtime.

Usage notes

- This keyword is ignored if DESTYPE is SCREEN.
- If DESTYPE is PREVIEW, Report Builder uses DESNAME to determine which printer’s fonts to use to display the output.
- The argument(s) for this keyword may be case sensitive, depending on your operating system.

In some cases, this parameter may be overridden by your operating system.
**DESFORMAT**

**Description** In bit-mapped environments, DESFORMAT specifies the printer driver to be used when DESTYPE is FILE. In character-mode environments, it specifies the characteristics of the printer named in DESNAME.

**Syntax**

```
[DESFORMAT=]desformat
```

**Values**

Any valid destination format not to exceed 1K in length. Examples of valid values for this keyword are, for example, hpl, hplwide, dec, decwide, decland, dec180, df1t, wide. Ask your System Administrator for a list of valid destination formats.

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF</td>
<td>Means that the report output will be sent to a file that can be read by a PDF viewer. PDF output is based upon the currently configured printer for your system. The drivers for the currently selected printer is used to produce the output; you must have a printer configured for the machine on which you are running the report.</td>
</tr>
<tr>
<td>HTML</td>
<td>Means that the report output will be sent to a file that can be read by an HTML 3.0 compliant browser (for example, Netscape 2.2).</td>
</tr>
<tr>
<td>HTMLCSS</td>
<td>Means that the report output sent to a file will include style sheet extensions that can be read by an HTML 3.0 compliant browser that supports cascading style sheets.</td>
</tr>
<tr>
<td>HTMLCSSIE</td>
<td>Means that the report output sent to a file will include style sheet extensions that can be read by Microsoft Internet Explorer 3.x.</td>
</tr>
<tr>
<td>RTF</td>
<td>Means that the report output will be sent to a file that can be read by standard word processors (such as Microsoft Word). Note that when you open the file in MS Word, you must choose View → Page Layout to view all the graphics and objects in your report.</td>
</tr>
<tr>
<td>DELIMITED</td>
<td>Means that the report output will be sent to a file that can be read by standard spreadsheet utilities, such as Microsoft Excel. Note that if you do not choose a delimiter, the default delimiter is a TAB.</td>
</tr>
<tr>
<td>XML</td>
<td>Means that the report output will be an XML document, saved as a separate file with the .XML extension. This report can be opened and read in an XML-supporting browser, or your choice of XML viewing application.</td>
</tr>
</tbody>
</table>
**Default**

Taken from the Initial Value property of the DESFORMAT parameter. For bit-mapped Report Builder, if DESFORMAT is blank or `dflt`, then the current printer driver (specified in **File → Choose Printer**) is used. If nothing has been selected in Choose Printer, then PostScript is used by default.

**Usage notes**

- This keyword is ignored if DESTYPE is SCREEN.
- The value or values for this keyword may be case sensitive, depending on your operating system.

**CACHELOB**

**Description** CACHELOB specifies whether to cache retrieved Oracle8 large object or objects in the temporary file directory (specified by REPORTS60_TMP).

**Values**

YES means to cache the LOB in the temporary file directory.

NO means to not cache the LOB in the temporary file directory.

**Default**

YES

**Usage notes**

- You can only set this option on the command line.
- If the location of the temporary file directory does not have sufficient available disk space, it is preferable to set this value to NO. Setting the value to NO, however, may decrease performance, as the LOB may need to be fetched from the server multiple times.
COPIES
Description COPIES is the number of copies of the report output to print.
Syntax

[COPIES=]n

Values
Any valid integer from 1 through 9,999.
Default
Taken from the Initial Value property of the COPIES parameter.
Usage notes
- This keyword is ignored if DESTYPE is not Printer.
- If COPIES is left blank on the Runtime Parameter Form, it defaults to one.

CURRENCY
Description CURRENCY is the currency character to be used in number formats.
Syntax

[CURRENCY=]currency_symbol

Values
Any valid alphanumeric string not to exceed 1K in length.
Default
The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default of up to four characters in the Initial Value property of the CURRENCY parameter.
Usage note
A CURRENCY value entered in Property Palette will override any CURRENCY value entered on the command line.
**THOUSANDS**

*Description* THOUSANDS is the thousands character to be used in number formats.

*Syntax*

```
[THOUSANDS=]thousands_symbol
```

*Values*

Any valid alphanumeric character.

*Default*

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default of up to four characters in the Initial Value property of the THOUSANDS parameter.

*Usage notes*

- A THOUSANDS value entered on the Parameter property sheet will override any THOUSANDS value entered on the command line.
- The alphanumeric character defined as the THOUSANDS value is the actual value that is returned. For example, if you define "," as the THOUSANDS value, "," is returned.

**DECIMAL**

*Description* DECIMAL is the decimal character to be used in number formats.

*Syntax*

```
[DECIMAL=]decimal_symbol
```

*Values*

Any valid alphanumeric character.

*Default*

The default for ORACLE is determined by the ORACLE National Language Support facilities. You can also set a default in the Initial Value property of the DECIMAL parameter.
Usage notes

- A DECIMAL value entered on the Parameter property sheet will override any DECIMAL value entered on the command line.
- The alphanumeric character defined as the DECIMAL value is actual value that is returned. For example, if you define "." as the DECIMAL value, "." is returned.

READONLY

Description
READONLY requests read consistency across multiple queries in a report. When accessing data from ORACLE, read consistency is accomplished by a SET TRANSACTION READ ONLY statement (refer to your Oracle8 Server SQL Language Reference Manual for more information on SET TRANSACTION READ ONLY).

Syntax

```
[READONLY=] {YES | NO}
```

Values

YES requests read consistency.

NO means do not provide read consistency.

Default

NO

Usage note

This keyword is only useful for reports using multiple queries, because ORACLE automatically provides read consistency, without locking, for single query reports.

Restrictions

- In the Report trigger order of execution, notice where the SET TRANSACTION READONLY occurs.
LOGFILE

Description LOGFILE is the name of the file to which File → Print Screen output is sent. If the specified file already exists, output will be appended to it. This keyword is only used in character mode.

Syntax

[LOGFILE=] logfile

Values

Any valid filename.

Default

dfltrep.log in the current directory.

BUFFERS

Description BUFFERS is the size of the virtual memory cache in kilobytes. You should tune this setting to ensure that you have enough space to run your reports, but not so much that you are using too much of your system’s resources.

Syntax

[BUFFERS=] n

Values

A number from 1 through 9,999. For some operating systems, the upper limit may be lower.

Default

640K

Usage note

If this setting is changed in the middle of your session, then the changes do not take effect until the next time the report is run.
BATCH

**Description** If BATCH is specified, it must be YES.

**Syntax**

```
[BATCH=]YES
```

PAGESIZE

**Description** PAGESIZE is the dimensions of the physical page (that is, the size of the page that the printer outputs). The page must be large enough to contain the report. For example, if a frame in a report expands to a size larger than the page dimensions, the report will not run.

**Syntax**

```
[PAGESIZE=]width x height
```

**Values**

Any valid page dimensions of the form: page width x page height, where page width and page height are zero or more. The maximum width/height depends upon the unit of measurement. For inches, the maximum width/height is 512 inches. For centimeters, it is 1312 centimeters. For picas, it is 36,864 picas.

**Default**

For bitmap, 8.5 x 11 inches. For character mode, 80 x 66 characters. If the report was designed for character mode and is being run or converted on bitmap, then the following formula is used to determine page size if none is specified:

\[
\text{(default page size} \times \text{character page size)}/\text{default character page size.}
\]

For example, if the character page size is 80 x 20, the bit-mapped page size would be: \((8.5 \times 80)/80 \times (11 \times 20)/66 = 8.5 \times 3.33.\)

**Usage notes**

- On some printers the printable area of the physical page is restricted. For example, the sheet of paper a printer takes may be 8.5 x 11 inches, but the printer might only be able to print on an area of 8 x 10.5 inches. If you define a page width x page height in Report Builder that is bigger than the printable area your printer allows, clipping may occur in your report output. To avoid clipping, you can either increase the printable area for the printer (if your operating system allows it) or you can set the page width x page height to be the size of the printable area of the page.
If this keyword is used, then its value overrides the page dimensions of the report definition.

A PAGESIZE value entered on the Runtime Parameter Form will override any PAGESIZE value entered on the command line.

PROFILE

Description PROFILE is the name of a file in which you want to store performance statistics on report execution. If you specify a filename, then Report Builder calculates statistics on the elapsed and CPU time spent running the report. PROFILE calculates the following statistics:

- **TOTAL ELAPSED TIME** is the amount of time that passes between when you issue RWBLD60 and when you leave the designer. TOTAL ELAPSED TIME is the sum of Report Builder Time and ORACLE Time.

- **Time** is the amount of time spent in Report Builder.

- **ORACLE Time** is the amount of time spent in the database and is composed of the following:
  - **UPI** is the amount of time spent to do such things as connect to the database, parse the SQL, and fetch the data.
  - **SQL** is the amount of time spent performing SRW.DO_SQL.
  - **TOTAL CPU Time used by process** is the CPU time spent while in the designer.

Note: For some operating systems, the Report Builder time includes the database time because the database is included in Report Builder's process.

Syntax

```
[PROFILE=]profiler_file
```

Values

Any valid filename in the current directory.
RUNDEBUG

Description
RUNDEBUG is whether you want extra runtime checking for logical errors in reports. RUNDEBUG checks for things that are not errors but might result in undesirable output. RUNDEBUG checks for the following:

- Frames or repeating frames that overlap but do not enclose another object. This can lead to objects overwriting other objects in the output.
- Layout objects with page-dependent references that do not have fixed sizing. Report Builder will make such objects fixed in size regardless of the Vertical and Horizontal Elasticity properties.
- Bind variables referenced at the wrong frequency in PL/SQL.

Syntax

[RUNDEBUG=] {YES | NO}

Values
YES means perform extra runtime error checking.
NO means do not perform extra runtime error checking.

Default
YES

ONSUCCESS

Description
ONSUCCESS is whether you want a COMMIT or ROLLBACK performed when a report is finished executing.

Syntax

[ONSUCCESS=] {COMMIT | ROLLBACK | NOACTION}

Values
COMMIT means perform a COMMIT when a report is done.
ROLLBACK means perform a ROLLBACK when a report is done.
NOACTION means do nothing when a report is done.

Default
COMMIT, if a USERID is provided.
NOACTION, if called from an external source (for example, Oracle Forms Developer) with no USERID provided.
Usage note
The COMMIT or ROLLBACK for ONSUCCESS is performed after the after report trigger fires. Other COMMITs and ROLLBACKs can occur prior to this one. For more information, see the READONLY command.

ONFAILURE
Description ONFAILURE is whether you want a COMMIT or ROLLBACK performed if an error occurs and a report fails to complete.

Syntax

[ONFAILURE=]{COMMIT|ROLLBACK|NOACTION}

Values
COMMIT means perform a COMMIT if a report fails.
ROLLBACK means perform a ROLLBACK if a report fails.
NOACTION means do nothing if a report fails.

Default
ROLLBACK, if a USERID is provided.
NOACTION, if called from an external source (for example, Oracle Forms Developer) with no USERID provided.

Usage note
The COMMIT or ROLLBACK for ONFAILURE is performed after the after fails. Other COMMITs and ROLLBACKs can occur prior to this one. For more information, see the READONLY command.

KEYIN
Description KEYIN is the name of a keystroke file that you want to execute at runtime. KEYIN is used to run the keystroke files created with KEYOUT. Since KEYIN is used to execute a keystroke file, it is only relevant when running in a character-mode environment.

Syntax

[KEYIN=}keyin_file

Values
Any valid key filename in the current directory.
KEYOUT

**Description**

KEYOUT is the name of a keystroke file in which you want Reports Runtime to record all of your keystrokes. You can then use KEYIN to execute the keystroke file. KEYOUT and KEYIN are useful when you have certain keystrokes that you want to do each time you run a report. They are also useful for debugging purposes. Since KEYOUT is used to create a keystroke file, it is only relevant when running reports in a character-mode environment.

**Syntax**

```
[KEYOUT=]keyout_file
```

**Values**

Any valid filename.

ERRFILE

**Description**

ERRFILE is the name of a file in which you want Report Builder to store error messages.

**Syntax**

```
[ERRFILE=]error_file
```

**Values**

Any valid filename.

LONGCHUNK

**Description**

LONGCHUNK is the size (in kilobytes) of the increments in which Report Builder retrieves a LONG column value. When retrieving a LONG value, you may want to retrieve it in increments rather than all at once because of memory size restrictions. LONGCHUNK applies only to Oracle7 and Oracle8.

**Syntax**

```
[LONGCHUNK=]n
```

**Values**

A number from 1 through 9,999. For some operating systems, the upper limit may be lower.

**Default**

10K
ORIENTATION

Description  ORIENTATION controls the direction in which the pages of the report will print.

Syntax

    [ORIENTATION=] {DEFAULT|LANDSCAPE|PORTRAIT}

Values

DEFAULT means use the current printer setting for orientation.
LANDSCAPE means landscape orientation
PORTRAIT means portrait orientation

Default

DEFAULT

Usage notes

- If ORIENTATION=LANDSCAPE for a character mode report, then you must ensure that your printer definition file contains a landscape clause.
- Not supported when output to a PCL printer on Motif.

BACKGROUND

Description  BACKGROUND is whether the call is synchronous (BACKGROUND=NO) or asynchronous (BACKGROUND=YES). A synchronous call means that the client waits for the report to queue, be assigned to a runtime engine, run, and finish. An asynchronous call means that the client simply sends the call without waiting for it to complete. If the client process is killed during a synchronous call, the job is canceled.

Syntax

    [BACKGROUND=] {YES|NO}

Values

YES or NO

Default

NO
MODE
Description MODE specifies whether to run the report in character mode or bitmap. This enables you to run a character-mode report from bit-mapped Report Builder or vice versa. For example, if you want to send a report to a PostScript printer from a terminal (for example, a vt220), you could invoke character-mode RWRUN60 and run the report with MODE=BITMAP. On Windows, specifying MODE=CHARACTER means that the Report Builder ASCII driver will be used to produce editable ASCII output.

Syntax

[MODE=] {BITMAP | CHARACTER | DEFAULT}

Values

BITMAP
DEFAULT means to run the report in the mode of the current executable being used.

CHARACTER
Default
DEFAULT

PRINTJOB
Description PRINTJOB specifies whether the Print Job dialog box should be displayed before running a report.

Syntax

[PRINTJOB=] {YES | NO}

Values

YES or NO
Default
NO
Usage notes

- When a report is run as a spawned process (that is, one executable, such as RWRUN60, is called from within another executable, such as RWBLD60), the Print Job dialog box will not appear, regardless of PRINTJOB.
- When DESTYPE=MAIL, the Print Job dialog box will not appear, regardless of PRINTJOB.

TRACEFILE

Description
TRACEFILE is the name of the file in which Report Builder logs trace information.

Syntax

[TRACEFILE=]tracefile

Values

Any valid filename.

Usage notes

- Trace information can only be generated when running an .RDF file. You cannot specify logging when running a .REP file.
- If you specify LOGFILE or ERRFILE as well as TRACEFILE, then all of the trace information will be placed in the most recently specified file. For example, in the following case, all of the specified trace information would be placed in the err.log because it is the last file specified in the RWRUN60 command.

    RWRUN60 MODULE=order_entry
    USERID=scott/tiger
    TRACEFILE=trace.log LOGFILE=mylog.log
    ERRFILE=err.log

TRACEMODE

Description
TRACEMODE indicates whether Report Builder should add the trace information to the file or overwrite the entire file.

Syntax

[TRACEMODE=]{TRACE_APPEND|TRACE_REPLACE}
Values
TRACE_APPEND adds the new information to the end of the file.
TRACE_REPLACE overwrites the file.

Default
TRACE_APPEND

Usage note
Trace information can only be generated when running an .RDF file. You cannot specify logging when running a .REP file.

TRACEOPTS
Description TRACEOPTS indicates the tracing information that you want to be logged in the trace file when you run the report.

Syntax

[TRACEOPTS=] (TRACE_ERR|TRACE_PRF|TRACE_APP|TRACE_PLS|TRACE_SQL|TRACE_TMS|TRACE_DST|TRACE_ALL| (opt1, opt2, ...))

Values
A list of options in parentheses means you want all of the enclosed options to be used. For example, TRACE_OPTS=(TRACE_APP, TRACE_PRF) means you want TRACE_APP and TRACE_PRF applied.

TRACE_ALL means log all possible trace information in the trace file.
TRACE_APP means log trace information on all the report objects in the trace file.
TRACE_BRK means list breakpoints in the trace file.
TRACE_DST means list distribution lists in the trace file. You can use this information to determine which section was sent to which destination. The trace file format is very similar to the .DST file format, so you can cut and paste to generate a .DST file from the trace file.
TRACE_ERR means list error messages and warnings in the trace file.
TRACE_PLS means log trace information on all the PL/SQL objects in the trace file.
TRACE_PRF means log performance statistics in the trace file.
TRACE_SQL means log trace information on all the SQL in the trace file.
TRACE_TMS means enter a timestamp for each entry in the trace file.

Default
TRACE_ALL

Usage note
Trace information can only be generated when running a .RDF file. You cannot specify logging when running a .REP file.

**AUTOCOMMIT**

Description Specifies whether database changes (for example, CREATE) should be automatically committed to the database. Note that some non-ORACLE databases (for example, SQL Server) require that AUTOCOMMIT=YES.

Syntax

```
[AUTOCOMMIT=} {YES|NO}
```

Values

YES or NO

Default

NO

**NONBLOCKSQL**

Description NONBLOCKSQL specifies whether to allow other programs to execute while Reports Runtime is fetching data from the database.

Syntax

```
[NONBLOCKSQL=} {YES|NO}
```

Values

YES means that other programs can execute while data is being fetched.
NO means that other programs cannot execute while data is being fetched.

Default

YES
**ROLE**

**Description** ROLE specifies the database role to be checked for the report at runtime. ROLE is ignored for RWBLD60.

**Syntax**

```
[ROLE=] {rolename/[rolepassword]}
```

**Values**

A valid role and (optionally) a role password.

**DISABLEPRINT**

**Description** DISABLEPRINT specifies whether to disable File → Print, or File → Choose Printer (on Motif) and the equivalent toolbar buttons in the Runtime Previewer.

**Syntax**

```
[DISABLEPRINT=] {YES|NO}
```

**Values**

YES or NO

**Default**

NO when there are blank pages in your report output that you do not want to print.

**DISABLEMAIL**

**Description** DISABLEMAIL specifies whether to disable the Mail menu and the equivalent toolbar buttons in the Runtime Previewer.

**Syntax**

```
[DISABLEMAIL=] {YES|NO}
```

**Values**

YES or NO

**Default**

NO
DISABLEFILE

Description DISABLEFILE specifies whether to disable the File → Generate to File menu in the Runtime Previewer.

Syntax

[DISABLEFILE=] {YES|NO}

Values
YES or NO

Default
NO

DISABLENEW

Description DISABLENEW specifies whether to disable the View → New Previewer menu to prevent the ability to display a new instance of the Runtime Previewer.

Syntax

[DISABLENEW=] {YES|NO}

Values
YES or NO

Default
NO

DESTINATION

Description The DESTINATION keyword allows you to specify the name of a .DST file that defines the distribution for the current run of the report.

Syntax

[DESTINATION=] filename.DST

Values
The name of a .DST file that defines a report or report section distribution.

Usage note
To enable the DESTINATION keyword, you must specify DISTRIBUTE=YES on the command line.
**DISTRIBUTE**

**Description**
DELIMITER specifies the character or characters to use to separate the cells in your report output.

DISTRIBUTE enables or disables distributing the report output to multiple destinations, as specified by the distribution list defined in the report distribution definition or a .DST file.

**Syntax**

```
[DISTRIBUTE=] {YES|NO}
```

**Values**

YES means to distribute the report to the distribution list.

NO means to ignore the distribution list and output the report as specified by the DESNAME and DESFORMAT parameters. This is fundamentally a debug mode to allow running a report set up for distribution without actually executing the distribution.

**Default**

NO

**Usage note**

To enable the DESTINATION keyword, you must specify DISTRIBUTE=YES.

---

**PAGESTREAM**

**Description**
PAGESTREAM enables or disables page streaming for the report when formatted as HTML or HTMLCSS output, using the navigation controls set by either of the following:

- PL/SQL in a Before Report trigger (SRW.SET_PAGE_NAVIGATION_HTML)

**Syntax**

```
[PAGESTREAM=] {YES|NO}
```
Values
YES means to stream the pages.
NO means to output the report without page streaming.
Default
NO

BLANKPAGES
Description BLANKPAGES specifies whether to suppress blank pages when you print a report. Use this keyword when there are blank pages in your report output that you do not want to print.
Syntax
[BLANKPAGES=] {YES|NO}
Values
YES means print all blank pages
NO means do not print blank pages
Default
YES
Usage note
BLANKPAGES is especially useful if your logical page spans multiple physical pages (or panels), and you wish to suppress the printing of any blank physical pages.

SERVER
Description SERVER is the TNS service entry name of the Reports Server.
Syntax
[SERVER=] tnsname
Values
Any valid TNS service entry name.
Usage note

If you set the REPORTS60_REPORTS_SERVER environment variable on your
Web Server machine, then you can omit the SERVER command line argument
to process requests using the default server, or you can include the SERVER
command line argument to override the default.

JOBNAME

Description JOBNAME is the name for a job to appear in the Reports Queue
Manager. It is treated as a comment and has nothing to do with the running of
the job. If it is not specified, then the queue manager will show the report name
as the job name.

Syntax

\[\text{JOBNAME=}\text{string}\]

SCHEDULE

Description SCHEDULE is a scheduling command. The default is now. To
eliminate the need for quoting the scheduling command, use underscore (_)
instead of a space. For example:

\begin{verbatim}
schedule=every_first_fri_of_month_from_15:53_Oct_23,_1999_retry_3_after_1_hour
schedule=last_weekday_before_15_from_15:53_Oct_23,_1999_retry_after_1_hour
\end{verbatim}

Note: Earlier forms of the SCHEDULE syntax are supported, but only the
current SCHEDULE syntax is documented here.

Syntax

\[\text{SCHEDULE=}\text{string}\]

where the string is:

\begin{verbatim}
[FREQ from] TIME [retry {n} + after LEN]
\end{verbatim}

<table>
<thead>
<tr>
<th>FREQ</th>
<th>hourly</th>
<th>daily</th>
<th>weekly</th>
<th>monthly</th>
<th>every LEN [DAYREPEAT]</th>
<th>last [WEEKDAYS</th>
<th>weekday</th>
<th>weekend] before [n]+</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEN</td>
<td>[n]+</td>
<td>[minute[s]]</td>
<td>hour[s]</td>
<td>day[s]</td>
<td>week[s]</td>
<td>month[s]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAYREPEAT</td>
<td>first</td>
<td>second</td>
<td>third</td>
<td>fourth</td>
<td>fifth</td>
<td>WEEKDAYS</td>
<td>month</td>
<td></td>
</tr>
<tr>
<td>WEEKDAYS</td>
<td>mon</td>
<td>tue</td>
<td>wed</td>
<td>thu</td>
<td>fri</td>
<td>sat</td>
<td>sun</td>
<td></td>
</tr>
</tbody>
</table>

RWCLI60 Command Line Arguments  A-193
TOLERANCE

**Description**  TOLERANCE is the time tolerance for duplicate job detection in minutes. TOLERANCE determines the maximum acceptable time for reusing a report’s cached output when a duplicate job is detected. Setting the time tolerance on a report reduces the processing time when duplicate jobs are found.

See Section 4.2, "Duplicate Job Detection" for more information on duplicate job detection.

**Syntax**

```plaintext
[TOLERANCE=]number
```

**Values**

Any number of minutes starting from 0

**Usage notes**

- If tolerance is not specified, then the Reports Server will rerun the report even if a duplicate report is found in the cache.
- If a report is being processed (that is, in the current job queue) when an identical job is submitted, then the Reports Server will reuse the output of the currently running job even if TOLERANCE is not specified or is set to zero.
**DELIMITER**

**Description**  
DELIMITER specifies the character or characters to use to separate the cells in your report output.

**Syntax**  
```
[DELIMITER=]value
```

**Values**  
Any alphanumeric character or string of alphanumeric characters, such as:

- ,  
  means a comma separates each cell
- .  
  means a period separates each cell

You can also use any of these four reserved values:

- tab  
  means a tab separates each cell
- space  
  means a space separates each cell
- return  
  means a new line separates each cell
- none  
  means no delimiter is used

You can also use escape sequences based on the ASCII character set, such as:

- \t  
  means a tab separates each cell
- \n  
  means a new line separates each cell

**Default**  
Tab

**Usage note**  
This argument can only be used if you have specified DESFORMAT=DELIMITED.
CELLWRAPPER

Description
CELLWRAPPER specifies the character or characters that displays around the delimited cells in your report output.

Syntax

[CELLWRAPPER=]value

Value

Any alphanumeric character or string of alphanumeric characters.

" means a double quotation mark displays on each side of the cell

' means a single quotation mark displays on each side of the cell

You can also use any of these four reserved values:

tab means a tab displays on each side of the cell
space means a single space displays on each side of the cell
return means a new line displays on each side of the cell
none means no cell wrapper is used

You can also use escape sequences based on the ASCII character set, such as:

\t means a tab displays on each side of the cell
\n means a new line displays on each side of the cell

Default

None

Usage notes

- This argument can only be used if you have specified DESFORMAT=DELIMITED.
- The cell wrapper is different from the actual delimiter.
DATEFORMATMASK
Description DATEFORMATMASK specifies how date values display in your delimited report output.

Syntax

\[\text{DATEFORMATMASK=}]\text{mask}\]

Values
Any valid date format mask

Usage note
This argument can only be used if you have specified DESFORMAT=DELIMITED.

NUMBERFORMATMASK
Description NUMBERFORMATMASK specifies how number values display in your delimited report output.

Syntax

\[\text{NUMBERFORMATMASK=}]\text{mask}\]

Values
Any valid number format mask

Usage note
This argument can only be used if you have specified DESFORMAT=DELIMITED.

EXPRESS_SERVER
Description EXPRESS_SERVER specifies the Express Server to which you want to connect.

Syntax

\text{EXPRESS\_SERVER=}\text{"server=server\}/domain=domain/}\text{user=userid\}/password=passwd\"

RWCLI60 Command Line Arguments   A-197
Syntax with RAM

`EXPRESS_SERVER="server=[server]/domain=[domain]/
user=[userid]/password=[passwd]/ramuser=[ramuserid]/
rampassword=[rampassword]/ramexpressid=[ramexpid]/
ramserverscript=[ramsscript]/rammasterdb=[ramdb]/
ramconnecttype=[ramconn]"`

Values

A valid connect string enclosed in double quotes ("" where

server is the Express server string (for example.,
ncacn_ip_tcp:olap2-pc/sl=x/st=x/ct=x/sv=x/). See below for more
details on the server string.

domain is the Express Server domain.

user is the userid to log on to the Express Server.

password is the password for the userid.

ramuser is the userid to log into the RDBMS.

rampassword is the password for the RDBMS.

ramexpressid is the Oracle Sales Analyzer database userid. This is required
for Oracle Sales Analyzer databases only.

ramserverscript is the complete file name (including the full path) of the
remote database configuration file (RDC) on the server. This file specifies
information such as the location of code and data databases. Using UNC
(Universal Naming Convention) syntax allows multiple users to use the same
connection to access the data without having to map the same drive letter to
that location. UNC syntax is \\ServerName\ShareName\ followed by any
subfolders and/or files.

rammasterdb is the name of the Relational Access Manager database to attach
initially. You must specify only the database file name. This database must
reside in a directory that is included in the path list in ServerDBPath for Express
Server. You can check the ServerDBPath in the File I/O tab of the Express
Configuration Manager dialog box.

ramconnecttype is the type of Express connection. Always specify 0 for a
direct connection.
Parameters

The server value contains four parameters that correspond to settings that are made in the Oracle Express Connection Editor and stored in connection (.XCF) files. All four parameters are required and can be specified in any order. The following table describes the parameters and their settings.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>sl</td>
<td>Server Login</td>
<td>-2: Host (Domain Login)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1: Host (Server Login)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No authentication required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Host (Domain Login) and Connect security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Host (Domain Login) and Call security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Host (Domain Login) and Packet security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: Host (Domain Login) and Integrity security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: Host (Domain Login) and Privacy security</td>
</tr>
<tr>
<td></td>
<td>Note: Windows NT uses all the settings. UNIX systems use only the settings 0, -1, and -2 See the Express Connection Editor Help system for information on these settings.</td>
<td></td>
</tr>
<tr>
<td>st</td>
<td>Server Type</td>
<td>:1: Express Server</td>
</tr>
<tr>
<td>ct</td>
<td>Connection Type</td>
<td>0: Express connection</td>
</tr>
<tr>
<td>sv</td>
<td>Server Version</td>
<td>1: Express 6.2 or greater</td>
</tr>
</tbody>
</table>

Usage notes

- You can have spaces in the string if necessary (for example, if the userid is John Smith) because the entire string is inside of quotes.

- If a forward slash (/) is required in the string, then you must use another forward slash as an escape character. For example, if the domain were tools/reports, then the command line should be as follows:

```
EXPRESS_SERVER="server=ncacn_ip_tcp:olap2-pc/sl=0/st=1/ct=0/sv=1/domain=tools/reports"
```

- You can use single quotes within the string. It is not treated specially because it is enclosed within double quotes.
**AUTHID**

**Description**

AUTHID is the username and password used to authenticate users to the restricted Reports Server. User authentication ensures that the users making report requests have access privileges to run the requested report. When users successfully log on, their browser is sent an encrypted cookie that authenticates them to the secured Reports Server registered in WebDB. By default, the cookie expires after 30 minutes. When a cookie expires, subsequent requests (that is, ones sent to a secured Reports Server) must be re-authenticated.

You can use the REPORTS60_COOKIE_EXPIRE environment variable to change the expiration time of the authentication cookie. See Appendix C, "Environment Variables" for more information.

If you want users to authenticate and remain authenticated until the cookie expires, then omit the AUTHID command line argument from the report request. If you want users to authenticate every time they run report requests, then use the Web CGI command SHOWAUTH and AUTHTYPE=S in the report URL, or include the %S argument to the key mapping entry in the cgicmd.dat (Web CGI) file.

**Syntax**

```
[AUTHID=]username/password
```

**Values**

Any valid username and password created in Oracle WebDB. See your DBA to create new users accounts in WebDB.

---

**CUSTOM**

**Description**

CUSTOMIZE specifies an XML file that you want to apply to the report when it is run. The XML file contains customizations (for example, font changes or color changes) that change the report definition in some way.

**Syntax**

```
[CUSTOMIZE=]filename.xml | (filename1.xml, filename2.xml, . . .)
```

**Values**

A filename or list of filenames that contain a valid XML report definition, with path information prefixed to the filename or filenames if necessary.
SAVE_RDF

Description SAVE_RDF specifies a file to which you want to save a combined .RDF file and .XML customization file. This argument is most useful when you have an .RDF file to which you are applying an .XML file via the CUSTOMIZE keyword and want to save the combination of the two to a new .RDF file.

Syntax

[SAVE_RDF=]filename.rdf

Values

Any valid filename.
Reports Server Configuration Parameters

This appendix contains a comprehensive list of Reports Server configuration parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACHEDIR</td>
<td>CACHEDIR is the cache for the Reports Server. CACHEDIR can be set to any directory or logical drive on the machine. If it is not specified, then the default is ORACLE_HOME\REPORT60\SERVER\CACHE.</td>
</tr>
<tr>
<td></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td></td>
<td>CACHEDIR= &quot;c:\ORACLE_HOME\Report60\cache&quot;</td>
</tr>
<tr>
<td>CACHESIZE</td>
<td>CACHESIZE is the size of the cache in megabytes. If you expect to store the output of many of your reports in the Reports Server cache, then you may want to increase this setting. If you do not expect to store a lot of output in the cache and have limited system resources, you may want to reduce it. Once the cache grows beyond the set size, the Reports Server cleans up the cached files on a first in, first out basis.</td>
</tr>
<tr>
<td></td>
<td><strong>Default value=50</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can set this parameter from the Queue Manager. Open the Queue Manager and log on as the administrator. Choose Queue → Properties, and then change the Cache size (MB) setting.</td>
</tr>
</tbody>
</table>
CLUSTERCONFIG is the configuration of slave servers to the master server. Clustering allows you to run reports on multiple Reports Servers. The master server can identify available slave servers and start their engines as needed. You can set up many servers as slaves to the master server. Use the following syntax in the master server configuration file:

```
Clusterconfig="(server=<servername>
  minengine=<minimum number of master engines>
  maxengine=<maximum number of master engines>
  initengine=<initial number of master engines>
  cachedir=<directory of central cache>)"
```

Note: Each slave definition must be enclosed in parentheses.

See Chapter 6, "Configuring Reports Server Clusters" for detailed instructions.

ENGLIFE is the maximum number of reports that an engine will run before shutting itself down. The Reports Server will then bring up fresh engines for new requests.

Default value=50

FAILNOTEFILE is path and file name of the notification message template that is sent to specified email addresses for jobs that fail to run.

Example

```
FAILNOTEFILE="c:\ORACLE_HOME\Report60\failnote.dat"
```

IDENTIFIER is an internal setting that contains the encrypted queue administrator user ID and password. You should not attempt to modify it. If IDENTIFIER is not specified or is deleted, or the configuration file is not present, then anyone can supply any user ID and password from the Reports Queue Manager to log on as the queue administrator. Once someone has logged on in this way, the user ID and password they specified becomes the queue administrator user ID and password until it is changed from the Queue Manager.

Table B–1  Reports Server configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUSTERCONFIG</td>
<td>CLUSTERCONFIG is the configuration of slave servers to the master server. Clustering allows you to run reports on multiple Reports Servers. The master server can identify available slave servers and start their engines as needed. You can set up many servers as slaves to the master server. Use the following syntax in the master server configuration file:</td>
</tr>
<tr>
<td>ENGLIFE</td>
<td>ENGLIFE is the maximum number of reports that an engine will run before shutting itself down. The Reports Server will then bring up fresh engines for new requests.</td>
</tr>
<tr>
<td>FAILNOTEFILE</td>
<td>FAILNOTEFILE is path and file name of the notification message template that is sent to specified email addresses for jobs that fail to run.</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>IDENTIFIER is an internal setting that contains the encrypted queue administrator user ID and password. You should not attempt to modify it. If IDENTIFIER is not specified or is deleted, or the configuration file is not present, then anyone can supply any user ID and password from the Reports Queue Manager to log on as the queue administrator. Once someone has logged on in this way, the user ID and password they specified becomes the queue administrator user ID and password until it is changed from the Queue Manager.</td>
</tr>
</tbody>
</table>
### Reports Server configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITENGINE</td>
<td>INITENGINE is the initial number of runtime engines started by the Reports Server. The server process will spawn this many engines when it is started. It will wait two minutes for these engines to connect to it and will shut itself down if they fail to do so. If the engines cannot connect in this amount of time, there is usually some setup problem. Default value=1</td>
</tr>
<tr>
<td>LOGOPTION</td>
<td>LOGOPTION is the type of log information you want inserted into the log file. The options are alljob, failedjob, and succeededjob.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>LOGOPTION=&quot;alljob&quot;</td>
</tr>
<tr>
<td>MAILPROFILE</td>
<td>(Windows NT only) If DESTYPE=MAIL, then Report Server sends your mail to a specific destination. MAILPROFILE allows you to specify the mail profile and password to be used when mailing reports from the Report Server.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>MAILPROFILE=&quot;mailprofileid/password&quot;</td>
</tr>
<tr>
<td>MAXCONNECT</td>
<td>MAXCONNECT is the maximum number of processes that can communicate with the server process at any one time. This setting is the sum of the number of engines and clients, and must be greater than two (at least one engine and one client). Default value=20</td>
</tr>
<tr>
<td>MAXENGINE</td>
<td>MAXENGINE is the maximum number of runtime engines available to the Reports Server to run reports. The server process will attempt to keep no more than this many engines active. Ensure that you have sufficient memory and resources available to accommodate this number of engines. Default value=1</td>
</tr>
</tbody>
</table>

*Note: You can set this parameter from the Queue Manager. Open the Queue Manager and log on as the administrator. Choose Queue → Properties, and then change the Simultaneous running engines Max setting.*
**Table B–1  Reports Server configuration parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIDLE</td>
<td>MAXIDLE is the maximum amount of time an engine is allowed to be idle before being shut down. Note that the Reports Server will not shut down the engine if doing so would reduce the number of available engines to less than those defined in the MINENGINE.</td>
</tr>
<tr>
<td></td>
<td>Default value=30</td>
</tr>
<tr>
<td></td>
<td>Note: You can set this parameter from the Queue Manager. Open the Queue Manager and log on as the administrator. Choose Queue ➔ Properties, and then change the Max idle time (minutes) before engine shutdown setting.</td>
</tr>
<tr>
<td>MINENGINE</td>
<td>MINENGINE is the minimum number of runtime engines the Reports Server should have available to run reports. The server process will attempt to keep at least this many engines active. Ensure that you have sufficient memory and resources available to accommodate this many engines.</td>
</tr>
<tr>
<td></td>
<td>Default value=0</td>
</tr>
<tr>
<td></td>
<td>Note: You can set this parameter from the Queue Manager. Open the Queue Manager and log on as the administrator. Choose Queue ➔ Properties, and then change the change the Simultaneous running engines Min setting.</td>
</tr>
<tr>
<td>PERSISTFILE</td>
<td>PERSISTFILE indicates the location of the Reports Server's .DAT file, which contains the details of scheduled jobs. If PERSISTFILE is not specified, then the default is ORACLE_HOME\REPORT60\SERVER.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>PERSISTFILE=&quot;C:\ORACLE_HOME\Report60\repserver.dat&quot;</td>
</tr>
<tr>
<td>REPOSITORYCONN</td>
<td>REPOSITORYCONN is the database connection string that connects the Reports Server to the database when the server starts up. The database takes a snapshot of Reports Server queue activity (that is, scheduled jobs) whenever jobs are run.</td>
</tr>
<tr>
<td></td>
<td>To create a queue activity table in your database, you must run rw_server.sql script.</td>
</tr>
<tr>
<td></td>
<td>For Web CGI configurations, see Section 3.2.6, &quot;Step 6. (Optional) Modifying the Reports Server configuration&quot;.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>REPOSITORYCONN=&quot;repserver_schema\password@mydb&quot;</td>
</tr>
</tbody>
</table>
SECURITY

SECURITY is the security level (0, 1, 2, or 3) for accessing cached output files through the Reports Queue Manager. 0 means that anyone can access a job’s cached output. 1 means that only a user whose user ID is identical to that of the user who ran the job can access the job’s cached output. 2 means that only the same process that sent the job can access the job’s cached output. 3 means that the cached output cannot be accessed.

Default value=1

SECURITYTNSNAME

SECURITYTNSNAME is the TNS name of the Oracle WebDB database that is used for authenticating users to the Reports Server. The Reports Server uses Oracle WebDB to perform a security check and to ensure that users have access privileges to run the report to the restricted Reports Server and, if requested, output to a restricted printer.

When the SECURITYTNSNAME parameter is set, you must add information about the Reports Server, printers, and reports in WebDB to process report requests via the Reports Server.

See Chapter 5, "Controlling User Access to Reports" for more information.

Example
SECURITYTNSNAME="sec_db"

SOURCEDIR

SOURCEDIR is a path to be searched before REPORTS60_PATH when searching for reports and other runtime files. This setting is useful when you have more than one Reports Server sharing the same ORACLE_HOME because each Reports Server can search different directories.

Example
SOURCEDIR="c:\my_reports"

SUCCNOTEFILE

SUCCNOTEFILE is the path and file name of the notification message template that is sent to specified email addresses for jobs that run successfully.

Example
SUCCNOTEFILE="c:\ORACLE_HOME\REPORT60\succnote.dat"
Table B–1  Reports Server configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| TEMPDIR   | TEMPDIR is a directory that will be used instead of REPORTS60_TMP when creating temporary files. TEMPDIR can be set to any directory or logical drive on the machine. **Example**
|           | TEMPDIR="c:\ORACLE_HOME\Report60\temp" |
This appendix contains detailed explanations of environment variables and configuration parameters that pertain to the Reports Server. See Table C–1, “Web CGI and Servlet environment variables” for a list of Web CGI and Servlet environments variables.

Environment variables are the configuration parameters used to control or customize the behavior of the Reports Server. For Windows NT, environment variables are set using the Registry Editor. For UNIX, variables can be set using a shell script.

*Table C–1  Web CGI and Servlet environment variables*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTS60_COOKIE_EXPIRE</td>
<td>Determines the expire time of the cookie in minutes.</td>
</tr>
<tr>
<td></td>
<td>Default value=30</td>
</tr>
<tr>
<td></td>
<td>Cookies save encrypted usernames and passwords on the client-side when users log on to a secured Reports Server to run report requests. When users successfully log on, their browser is sent an encrypted cookie. When a cookie expires, subsequent requests (that is, ones that are sent to secured Reports Servers), users must re-authenticate to run the report.</td>
</tr>
<tr>
<td>REPORTS60_DB_AUTH</td>
<td>Specifies the database authentication template used to log on to the database.</td>
</tr>
<tr>
<td></td>
<td>Default value=dbauth.htm</td>
</tr>
<tr>
<td>REPORTS60_ENCRYPTION_KEY</td>
<td>Specifies the encryption key used to encrypt the username and password for the cookie. The encryption key can be any character string.</td>
</tr>
<tr>
<td></td>
<td>Default value= reports6.0</td>
</tr>
</tbody>
</table>
Table C–1  Web CGI and Servlet environment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTS60_CGIDIAGBODYTAGS</td>
<td>For the Reports Web CGI, specifies HTML tags that will be inserted as a <code>&lt;BODY...&gt;</code> tag in the RWCGI60 diagnostic/debugging output. For example, you may want to use this environment to set up text/background color or image.</td>
</tr>
<tr>
<td>REPORTS60_CGIDIAGHEADTAGS</td>
<td>For the Reports Web CGI, specifies HTML tags to insert between <code>&lt;HEAD&gt;...&lt;/HEAD&gt;</code> tags in the RWCGI60 diagnostic/debugging output. For example, you may want to use this environment to set up <code>&lt;TITLE&gt;</code> or <code>&lt;META...&gt;</code> tags.</td>
</tr>
<tr>
<td>REPORTS60_CGIHELP</td>
<td>For the Reports Web CGI, defines URL/URI of the RWCGI60 help file, which will be navigated to when RWCGI60 is invoked with the empty request: <code>http://your_webserver/rwcgi60?</code>. For example, setting it to <code>http://www.yahoo.com</code> will go to that URL; setting it to <code>myhelpfile.htm</code> will display the file <code>http://your_webserver/myhelpfile.htm</code>. If this parameter is not defined, then a default help screen will be displayed.</td>
</tr>
<tr>
<td>REPORTS60_CGIMAP</td>
<td>For the Reports Web CGI, defines fully qualified filename/location of the RWCGI60 map file if map file configuration is used, (for example, <code>C:\ORANT\REPORT60\cgicmd.dat</code>).</td>
</tr>
<tr>
<td>REPORTS60_CGINODIAG</td>
<td>For the Reports Web CGI, when defined, disables all debugging/diagnostic output, such as help and showmap, from RWCGI60. For example, <code>http://your_webserver/rwcgi60/help?</code> will not work when REPORTS60_CGINODIAG is defined.</td>
</tr>
<tr>
<td>REPORTS60_REPORTS_SERVER</td>
<td>Specifies the default Reports Server for Web CGI requests. When this environment variable is set, you can omit the SERVER command line argument in report requests to process them using the default server, or you can include the SERVER command line argument to override the default.</td>
</tr>
<tr>
<td>REPORTS60_SSLPORT</td>
<td>If you are using SSL and you want to use a port number other than 443, you can use this variable to set a different port number. Default value= 443</td>
</tr>
</tbody>
</table>
### Table C–1  Web CGI and Servlet environment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTS60_SYS_AUTH</td>
<td>Specifies the authentication template used to authenticate the username and password when users run report request to a restricted Reports Server.</td>
</tr>
</tbody>
</table>
This appendix lists typical database connection strings that you or users can use when specifying report requests using the Web CGI or Servlet. A database connection string is the value used in the USERID command line argument to connect to the database.

See Appendix A, "RWCLI60 Command Line Arguments" for more information about the USERID command line argument.

### Table D–1 Database connection strings

<table>
<thead>
<tr>
<th>Database Connection String</th>
<th>Reports Server response</th>
<th>User action</th>
</tr>
</thead>
<tbody>
<tr>
<td>No USERID specified</td>
<td>Returns the database authentication form.</td>
<td>Types the Oracle or placeholder username and password.</td>
</tr>
<tr>
<td>Oracle username@database</td>
<td>Looks for the Oracle username and database pair in the connection string table to get the password. If the Reports Server finds the password, the report is run. If the password cannot be found, the Reports Server returns the database authentication form.</td>
<td>none</td>
</tr>
<tr>
<td>Oracle username/password@database</td>
<td>Accepts the connection string and runs the report.</td>
<td>none</td>
</tr>
<tr>
<td>Database Connection String</td>
<td>Reports Server response</td>
<td>User action</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Oracle username/password</td>
<td>Uses the local database and runs the report.</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>If there is no local database, then the Reports Server returns the database authentication form.</td>
<td>Types the Oracle database.</td>
</tr>
<tr>
<td>&lt;$username&gt;@database</td>
<td>Looks for the placeholder username in the connection string table. If the username cannot be found, the Reports Server returns the database authentication form.</td>
<td>Types the Oracle username and password.</td>
</tr>
<tr>
<td></td>
<td>If the Reports Server can find the placeholder username in the table, it looks for the Oracle username and database name pair in the table to get the password. If the Reports Server finds the password, the report is run.</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>If the password cannot be found in the table, the Reports Server returns the database authentication form.</td>
<td>Types the database password.</td>
</tr>
<tr>
<td>&lt;$username&gt;/password@database</td>
<td>Looks for the placeholder username in the connection string table. If the username is found, the Reports Server runs the report.</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>If the placeholder username cannot be found, it returns the database authentication form. The user must authenticate to run the report.</td>
<td>Types the Oracle username and password.</td>
</tr>
</tbody>
</table>
E.1 Reports Web Cartridge Example

This section contains step-by-step instructions on how to manually configure the Reports Server using the Web Cartridge on the Oracle Applications Server (OAS). In order to make these configuration procedures meaningful, it is necessary to make several assumptions. For the purposes of this section, the following assumptions are made:

- You are configuring the Reports Server to enable Web requests.
- You are using the Reports Web Cartridge with Oracle Application Server, Release 4.0 or later.
- The Reports Server is installed on a different machine than the Oracle Application Server.

E.1.1 Step 1. Install the software

You must have the Oracle Reports Server software to install and configure the Reports Server. Refer to the Getting Started manual for more information on installing the Oracle Reports Thin Client and Oracle Reports Server components and the Report Builder.

Tip: To install just the Reports Thin Client, Reports Server, or the Report Builder, start the Oracle Installer and choose the Custom Installation options. From the Available Products list box, expand the Oracle Reports Developer node and choose Reports Thin Client, Oracle Reports Server, or Report Builder as desired. Proceed with the installation as described in the Getting Started manual.
On the Web server machine:

1. Install the Oracle Reports Thin Client component after installing your Web server. Refer to your Web server documentation for more information on installing the Web server.

On the Reports Server machine:

1. Install the Oracle Reports Server component and the Report Builder. Note that the Report Builder is required only if you need to build reports. If the Reports Server machine is used only as a deployment machine, then do not install the Report Builder component.

2. When you install the Oracle Reports Server, you will be asked to identify the Reports Server TNS name and Port number. This information is added to the tnsnames.ora file (located in the ORACLE_HOME\NET80\ADMIN directory).
   - Reports Server TNS Name. Type repserver.world or keep the default. where repserver.world is the name of the server instance and .world is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.
   - Port. Type the port number to which the server will listen or keep the default.

   Tip: You can always edit the Reports Server service entry by opening the tnsnames.ora file in a text editor.

3. Make note of the Reports Server TNS name and port number. You will need this information to add the service entry for this Reports Server to the tnsnames.ora file located on the Web server machine. See Section E.1.3.1, "To create a service entry for the Reports Server" for more information.
E.1.2 Step 2. Configure your Web server

This step is performed on your Web server machine.

To configure the Oracle Application Server:
For more information on these steps, refer to your Oracle Application Server documentation.

1. Start the OAS Node Manager.
2. Start your browser.
3. Point your browser to the OAS Node Manager port. Expand the site and HTTP listeners node.
4. Create a listener if necessary, or expand the listener you want to use.

E.1.3 Step 3. Configure the Reports Web Cartridge

This step is performed on your Web server machine.

E.1.3.1 To create a service entry for the Reports Server

If the Web server is on a different machine than your Reports Server, then you must add the Reports Server service entry that was created on the Reports Service machine to the tnsnames.ora file located on your Web server machine. This will enable the Web CGI executable to communicate with the Reports Server.

Tip: If you do not remember the service entry settings for the Reports Server. Open the tnsnames.ora file located in ORACLE_HOME\NET80\ADMIN directory on your Reports Server machine. Copy or make note of the service entry.

1. On your Web Server machine, open the tnsnames.ora file (located in the ORACLE_HOME\NET80\ADMIN directory) in a text editor.
2. Add the following Reports Server service entry:

   repserver.world = (ADDRESS = (PROTOCOL = TCP)(Host = repserver_machine.mydomain)(Port = 1949))

   where:

   repserver.world is the name of the server instance and .world is the domain specified in the NAMES.DEFAULT_DOMAIN setting in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.
E.1.3.2 To set the default Reports Server

For Windows NT
1. On your desktop, choose Start→Run.
2. Type regedit to display the Registry Editor.
3. Expand Hkey_Local_machine→Software→Oracle.
4. Choose Edit→New→String value to add the following environment variable:

```
REPORTS60_REPORTS_SERVER=repserver
```

where repserver is the name of the Reports Server that you are configuring
(that is, the TNS service entry name of the Reports Server). See Section E.1.1,
"Step 1. Install the software".

Note: Refer to your operating system’s documentation to determine if the
values you set are case-sensitive.

For UNIX
Tip: You may want to create a shell script that sets environment variables on your
Web server machine. To do this, create a file that contains the command described
below.
1. Set the REPORTS60_REPORTS_SERVER environment variable to the name of
the Reports Server:

```
setenv REPORTS60_REPORTS_SERVER repserver
```

where repserver is the name of the Reports Server that you are configuring
(that is, the TNS service entry name of the Reports Server). See Section E.1.1,
"Step 1. Install the software".

Tip: Refer to your operating system’s documentation to determine if the values
you set are case-sensitive.
E.1.3.3 To configure the Reports Web Cartridge (on Windows NT and UNIX)
The steps that follow assume that you have already configured Oracle Application Server, Release 4.0 or later. For more information, refer to your Oracle Application Server documentation.

1. In the Oracle Application Server Welcome page, click OAS Manager.
2. In the OAS Manager navigational tree, click the + icon beside the web site icon.
3. Click the Applications icon.
4. Click the Add icon to display the Add Application form.
5. In the Add Application form, Application Type field, specify C Web.
6. Click Apply.
7. In the Add form, Application Name field, specify the name of your application (for example, Reports).
8. In the Display Name field, specify the display name (for example, Oracle Reports).
9. In the Application Version field, specify a version number (for example, 6.0).
10. Click Apply.
11. In the Success dialog, click Add Cartridge to this Application.
12. In the Add C Web Cartridge form, Cartridge Name field, type: Reports60Cartridge.
13. In the Display Name field, type: Reports60Cartridge.
14. In the Cartridge Shared Object field, specify the location of the shared object file (rwows60.dll). For example, type: D:\ORACLE_HOME\BIN\RWOWS60.DLL.
15. In the Entry Point (Shared Object) field, type: rwows_start.
16. In the Virtual Path field, type: /rwows.
17. Click Apply.
18. In the Success dialog box, click OK.
19. Within the Cartridge Configuration folder, click Tuning.
20. In the Maximum Number of Instances field, type: 100.
21. Click Apply.
22. In the Success dialog box, click OK.
E.1.4 Step 4. Configure the Reports Server

You can configure the Reports Server on Windows NT as a service or a non-service, or you can configure the Reports Server on UNIX. To configure the Reports Server, you need to create a TNS service name in the tnsname.ora file, set environment variables, and then install the Reports Server.

RWMTS60 is the executable for the Reports Server. It has three commands associated with it:

- **Install**, enables you to install the Reports Server as a service on Windows NT. See Section E.1.4.2.2, "Install the Reports Server on Windows NT as a service" for Windows NT. For UNIX, see Section E.1.4.3, "To configure the Reports Server on UNIX".

- **Listen**, enables you to install the Reports Server on Windows NT as a non-service. See Section E.1.4.2.3, "Configure the Reports Server on Windows NT as a non-service".

- **Uninstall**, enables you to uninstall the Reports Server on Windows NT as a service. See Section E.1.4.2.4, "Uninstall the Reports Server on Windows NT".

For Windows NT, you can install, listen, and uninstall the Reports Server manually or in batch mode. Set the BATCH argument to Yes to ignore any error messages raised during the installation and allow the operation to complete without interruption. Any errors or status information raised during installation can be viewed in the Reports Server log file (located on ORACLE_HOME\REPORT60\SERVER\repserver.log). The default for BATCH is No.

This step is performed on the Reports Server machine.

E.1.4.1 To configure the Reports Server on Windows NT

E.1.4.1.1 (Optional) Set the source path

Alternatively, after the Reports Server is installed, you can set the source path in the Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, "Reports Server Configuration Parameters" for more information.

1. Create the source directory if it does not already exist.

2. On your desktop, choose Start→Run.

3. Type regedit to display the Registry Editor.

4. Expand Hkey_Local_machine→Software→Oracle.
5. Double-click REPORTS60_PATH.

6. Add a new reports source path (for example, C:\WEB_REPORTS).
   
   **Note:** Refer to your operating system’s documentation to determine if the values you set are case-sensitive.

### E.1.4.2 To configure the Reports Server on Windows NT

#### E.1.4.2.1 (Optional) Set the source path

Alternatively, after the Reports Server is installed, you can set the source path in the Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, "Reports Server Configuration Parameters" for more information.

1. Create the source directory if it does not already exist.
2. On your desktop, choose **Start → Run**.
3. Type `regedit` to display the Registry Editor.
4. Expand `Hkey_Local_machine → Software → Oracle`.
5. Double-click REPORTS60_PATH.
6. Add a new reports source path (for example, C:\WEB_REPORTS).
   
   **Note:** Refer to your operating system’s documentation to determine if the values you set are case-sensitive.

#### E.1.4.2.2 Install the Reports Server on Windows NT as a service

1. On the Reports Server machine desktop, choose **Start → Run**.
2. Type the following command line argument:
   ```
   rwmts60 -install repserver tcpip
   ```
   or for batch mode:
   ```
   rwmts60 -install repserver batch=yes
   ```
   
   **Tip:** Repserver does not need to have the domain qualifier appended to it (for example, .world).
3. Click **Yes** to install the Reports Server as an NT Service and Click **OK** to start up the service.
4. On your desktop, choose Start → Settings → Control Panel and double-click (Services).

5. In the Services dialog box, choose Oracle Reports Server [repserver] and click Startup.

6. From the startup dialog, select This Account in the Log On As section and type or select an operating system user name and password. This specifies that the server is run as that user.

   Tip: If you want to output to PostScript or to a printer, then ensure the user running the Reports Server service has access to a default printer. Typically, the System Account does not usually have access to printers.

7. Set the service to start automatically (when the system is started).

8. Click OK.

9. Click Start. A Service Control message box indicates when your Reports Server has started. If your Reports Server cannot start, see Appendix F, "Troubleshooting" for more information.

   Tip: When you start the Reports Server for the first time, a Reports Server configuration file (for example, repserver.ora) is created in the ORACLE_HOME\REPORT60\SERVER directory. The setting for your Reports Server cache is set by default. You can change the cache directory, or set the report’s source path by modifying the configuration file. If you modify the configuration file, stop and restart the Reports Server for the changes to take effect.

E.1.4.2.3 Configure the Reports Server on Windows NT as a non-service

1. Run the following command line:

   rwmts60 -listen repserver

   or in batch mode:

   rwmts60 -listen repserver batch=yes

   Tip: Repserver does not need to have the domain qualifier (for example, .world) appended to it.
E.1.4.2.4 Uninstall the Reports Server on Windows NT

1. On the Reports Server machine desktop, choose **Start → Run**.

2. Type the following command line argument:
   
   rwmts60 -uninstall repserver
   
   or in batch mode:
   
   rwmts60 -uninstall repserver batch=yes

   **Tip:** Repserver does not need to have the domain qualifier appended to it (for example, `.world`).

E.1.4.3 To configure the Reports Server on UNIX

E.1.4.3.1 (Optional) Set environment variables

**Tip:** You may want to create a shell script that sets the environment variables and launches the RWMTS60 executable. To do this, create a file that contains the commands described below.

1. Create a directory for your source reports (for example, `/WEB_REPORTS`).

2. Set the **REPORTS60_PATH** environment variable to locate the reports:
   
   ```
   setenv REPORTS60_PATH $WEB_REPORTS
   ```

   **Tip:** Alternatively, after the Reports Server is installed, you can set the source path in the Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, "Reports Server Configuration Parameters" for more information.

3. Set the **TNS_ADMIN** environment variable to point to the location of the `tnsnames.ora` file:
   
   ```
   setenv TNS_ADMIN $ORACLE_HOME/NET80/ADMIN
   ```

   **Tip:** Refer to your operating system’s documentation to determine if the values you set are case-sensitive.
E.1.4.3.2 Starting the Reports Server

1. From the $ORACLE_HOME/BIN directory, run the following command line to run the Reports Server in the foreground:

   `rwmts60 name=repserver`

   or to run the Reports Server in the background:

   `rwmts60 name=repserver &`

   To determine if your server is running, go to step 6. If your Reports Server does not start or you get an error message, then refer to Appendix F, "Troubleshooting" for more information.

   **Tip:** When you start the Reports Server for the first time, a Reports Server configuration file (for example, `repserver.ora`) is created in the $ORACLE_HOME/REPORT60/SERVER directory. The setting for your Reports Server cache is set by default. If you want to change the cache directory, or if you want set the report's source path, then you can do so by modifying the configuration file. If you modify the configuration file, then you will need to stop and restart the Reports Server to acknowledge the changes.

2. (Optional) From the $ORACLE_HOME/BIN directory, run the following command line to ensure that the Reports Server is running:

   `rwrqv60 server=repserver`

   Status columns (that is, NAME, OWNER, DEST, etc.) for this Reports Server are displayed. Currently, though, no status information is available since no jobs are running.

   **Tip:** To kill the Reports Server, press ctrl-C.

   **Tip:** If you want to output to PostScript or to a printer, then the printer must be configured in the `uiprint.txt` file (located in the $ORACLE_HOME/GUICOM6/TK60/ADMIN directory).
E.1.5 Step 5. Run a report request

1. Ensure that the Reports Server is configured properly. In a Web browser make the following request:

   \[ \text{http://your_webserver/rwows60?report=your_report.rdf+userid=username/password@mydb+desformat=html+destype=cache} \]

   Notice that the SERVER command line argument is missing from the request. It is not required if you set the REPORTS60REPORTS_SERVER environment variable on your Web Server machine to the repserver.

   **Tip:** If the report does not run or if you receive an error message, then refer to Appendix F, "Troubleshooting" for more information.

This completes the Reports Server configuration example for a Web Cartridge configuration. You have successfully configured the Reports Server for dynamic reporting using the Reports Web Cartridge.

Next steps:

- (Optional) Tune the Reports Server to optimize performance or implement additional features, such as access control. See Section E.1.6, "Step 6. (Optional) Modify the configuration".
- Make reports available to users. See Chapter 4, "Running Report Requests" for more information on how to specify run requests and make them available to users.

E.1.6 Step 6. (Optional) Modify the configuration

This Reports Server is configured to perform using a default configuration. At some point, you may want to modify the configuration to tune performance, set up monitoring controls, or implement additional features.

- To update the database with job queue information, see Section E.1.6.1, "To update the database with job queue activity".
- To control user access to reports, see Chapter 5, "Controlling User Access to Reports".
- To configure the Reports Server for loading balancing, see Chapter 6, "Configuring Reports Server Clusters".
- To modify a report at runtime based on the audience, see Chapter 7, "Customizing Reports at Runtime".
■ To modify the Reports Server configuration file, see Appendix B, "Reports Server Configuration Parameters" for more information about the available configuration parameters.

■ Modify environment variables, see Section E.2, "Environment Variables".

E.1.6.1 To update the database with job queue activity
You can set up your database to take snapshots of the Reports Server queue activity whenever jobs are run. When you start the Reports Server, a connection to the database is made. By default, the Reports Server calls an API to delete queue information when the server restarts and to update the queue information in the database table.

You can edit the source for the API in the rw_server.sql script to override the defaults (for example, prevent the queue from being deleted when restarting the Reports Server). The prototype of the procedure (that is, the procedure name and the parameters it expects) should not be edited.

To connect to the database, you set the REPOSITORYCONN parameter in the configuration file. To create the queue activity table in the database, you run the rw_server.sql script.

On the Reports Server machine:

1. Open the repserver.ora configuration file (located on the ORACLE_HOME\REPORT60\SERVER directory) in a text editor.

2. Add the following configuration parameter:

   REPOSITORYCONN="repserver_schema/password@database"

   where repserver_schema/password@database is the connection string to the schema in your database that takes snapshots of queue activity of the specified Reports Server. In this case, repserver_schema is the schema for repserver queue activity.

   Tip: If you want to take snapshots of queue activity from many Reports Servers, then it is recommended that you create a different schema in your database for each Reports Server that you want to take snapshots of. This will prevent you from losing queue activity data when you restart a Reports Server.
3. **Run the rw_server.sql script** (located in the ORACLE_HOME\REPORT60\SQL directory) from the command line. At the command line prompt, type:

```
cd C:\ORACLE_HOME\REPORT60\SQL <RETURN>
plus80 username/password@my_db <RETURN>
@rw_server.sql <RETURN>
quid <RETURN>
```

4. **Stop and restart the Reports Server** to accept the changes made to the configuration file. When the Reports Server starts up, it connects to the database.

**Note:** When you restart your Reports Server, queue activity in the database is deleted by default. You can override the default by editing the API.

### E.2 Environment Variables

Environment variables are the configuration parameters used to control or customize the behavior of the Reports Server. For Windows NT, environment variables are set using the Registry Editor. For UNIX, variables can be set using a shell script.

#### Table E–1 Web Cartridge environment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTS60_COOKIE_EXPIRE</td>
<td>Determines the expire time of the cookie in minutes.</td>
</tr>
<tr>
<td></td>
<td>Default value=30</td>
</tr>
<tr>
<td></td>
<td>Cookies save encrypted usernames and passwords on the client-side when users log on to a secured Reports Server to run report requests. When users successfully log on, their browser is sent an encrypted cookie. When a cookie expires, subsequent requests (that is, ones that are sent to secured Reports Servers), user must re-authenticate to run the report.</td>
</tr>
<tr>
<td>REPORTS60_DB_AUTH</td>
<td>Specifies the database authentication template used to log on to the database.</td>
</tr>
<tr>
<td></td>
<td>Default value=dbauth.htm</td>
</tr>
</tbody>
</table>
**Table E–1  Web Cartridge environment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTS60_ENCRYPTION_KEY</td>
<td>Specifies the encryption key used to encrypt the user name and password for the cookie. The encryption key can be any character string.</td>
</tr>
<tr>
<td></td>
<td>Default value= reports6.0</td>
</tr>
<tr>
<td>REPORTS60_OWSDIAGBODYTAGS</td>
<td>For the Reports Web Cartridge, specifies HTML tags that will be inserted as a <code>&lt;BODY...&gt;</code> tag in the RWOWS diagnostic/debugging output. For example, you may want to use this environment to set up text/background color or image.</td>
</tr>
<tr>
<td>REPORTS60_OWSDIAGHEADTAGS</td>
<td>For the Reports Web Cartridge, specifies HTML tags to insert between <code>&lt;HEAD&gt; ...&lt;/HEAD&gt;</code> tags in the RWOWS diagnostic/debugging output. For example, you may want to use this environment to set up <code>&lt;TITLE&gt;</code> or <code>&lt;META...&gt;</code> tags.</td>
</tr>
<tr>
<td>REPORTS60_OWSHELP</td>
<td>For the Reports Web Cartridge, defines URL/URI of the RWOWS help file, which will be navigated to when RWOWS is invoked with the empty request: http://your_webserver/rwows?. For example, setting it to <a href="http://www.yahoo.com">http://www.yahoo.com</a> 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.</td>
</tr>
<tr>
<td>REPORTS60_OWSMAP</td>
<td>For the Reports Web Cartridge, defines fully qualified filename/location of the RWOWS map file if a map file is used, (for example, C:\ORANT\REPORT60\OWSCMD.DAT).</td>
</tr>
<tr>
<td>REPORTS60_OWSNODIAG</td>
<td>For the Reports Web Cartridge, when defined, disables all debugging/diagnostic output, such as help and showmap, from RWOWS. For example, http://your_webserver/rwows/help? will not work when REPORTS60_OWSNODIAG is defined.</td>
</tr>
<tr>
<td>REPORTS60_REPORTS_SERVER</td>
<td>Specifies the default Reports Server for Web Cartridge requests. When this parameter is set, you can omit the SERVER command line argument in report requests to process them using the default server, or you can include the SERVER command line argument to override the default.</td>
</tr>
</tbody>
</table>
### Table E-1  Web Cartridge environment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| REPORTS60_SSLPORT | If you are using SSL and you want to use a port number other than 443, you can use this variable to set a different port number.  
  Default value= 443                                                                                                                             |
| REPORTS60_SYS_AUTH | Specifies the authentication template used to authenticate the username and password when users run report request to a secured Reports Server.  
  Default value=sysauth.htm                                                                                                                   |
Troubleshooting

This appendix contains information on how to troubleshoot your Reports Server configuration.

Table F–1  Troubleshooting information

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Probable Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports Server appears to hang when you start it.</td>
<td>You may have made a syntactical error in the tnsnames.ora file and the Reports Server cannot resolve the TNS name. Alternatively, you could try rebooting in case the cause is a memory problem.</td>
</tr>
<tr>
<td>You get the error “Daemon failed to listen to port.”</td>
<td>If you start up a Reports Server that is listening to the same port as an already running Reports Server, then you will get this error. It could also be a problem with your Net8 or TCP/IP setup.</td>
</tr>
<tr>
<td>You get an error about being unable to initialize the printer (REP-3002)</td>
<td>Make sure that the Reports Server has access to printers. For Windows NT, the System Account does not usually have access to printers. It could be that you installed the Reports Server as an NT service and used the System Account or another account without printer access in the Log On As field. You must specify an account in the Log On As field that has a default printer access. This printer does not have to exist, but the driver must be installed. For UNIX, configure the printer in the uiprint.txt file.</td>
</tr>
</tbody>
</table>
Upon starting the Reports Server, you get server specific error 186.

Typically this indicates a problem in tnsnames.ora or sqlnet.ora. Check the entry for the Reports Server in tnsnames.ora. A typical entry should look something like the following:

```
repserver.world = (ADDRESS=(PROTOCOL=tcp)
(HOST=144.25.87.182) (PORT=1951))
```

In this example .world is appended to the name because it is the domain specified in the sqlnet.ora file. If the NAMES.DEFAULT_DOMAIN setting is not defined in the sqlnet.ora, then omit .world from the name of the server instance.

If your tnsnames.ora file appears to be correct, then check your sqlnet.ora file. Good default settings to use in this file are:

```
TRACE_LEVEL_CLIENT=OFF
names.directory_path = (TNSNAMES)
names.default_domain = world
name.default_zone = world
```

If your protocol is TCP, then ensure the Net8 TCP/IP adapter and Net8 have been installed. Lastly, be sure that your installed version of Net8 is not older than the version that came with the Reports Server.

Error reported when opening the report.

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→Options in the Explorer window.) This prevents you from creating files with names like your_report.rdf.txt. Alternatively, use a DOS window for file manipulation.

Alternatively, ensure the report is located in the path defined by the REPORTS60_PATH environment variable.

---

Table F–1  Troubleshooting information

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Probable Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upon starting the Reports Server, you get server specific error 186.</td>
<td>Typically this indicates a problem in tnsnames.ora or sqlnet.ora. Check the entry for the Reports Server in tnsnames.ora. A typical entry should look something like the following:</td>
</tr>
<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→Options in the Explorer window.) This prevents you from creating files with names like your_report.rdf.txt. Alternatively, use a DOS window for file manipulation. Alternatively, ensure the report is located in the path defined by the REPORTS60_PATH environment variable.</td>
</tr>
</tbody>
</table>
### Table F–1  Troubleshooting information

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Probable Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems running the Reports Server as a Windows NT Service.</td>
<td>If you install the Reports Server service to run under a user other than SYSTEM, then ensure the user account:</td>
</tr>
<tr>
<td></td>
<td>■ Has the Password Never Expires option selected in the User Manager.</td>
</tr>
<tr>
<td></td>
<td>■ Has membership in the appropriate groups to run the Reports Server and access the report files.</td>
</tr>
<tr>
<td></td>
<td>■ Has at least print permission to a default printer.</td>
</tr>
<tr>
<td></td>
<td>■ Can log on to a service. Choose Start → Programs → Administrative Tools → User Manager, then Policies User Rights. Check Show Advanced User Rights. From the Right list, choose Log on as a service. If the user is not already in the Grant To list, click the Add.</td>
</tr>
<tr>
<td></td>
<td>When starting the service, you may need to explicitly specify the domain as well as the user name (username/domain). If you get a Windows NT error reporting that the service failed and returning the error message number, then you can look up the message number in the Report Builder online help.</td>
</tr>
<tr>
<td>ops$ account is not working.</td>
<td>For security reasons, ops$ accounts are not supported by the Reports Server. If you pass a command line with USERID=.ops$ to the Reports Server, then an error is generated because it will try to use the user name of the Reports Server process rather than the user name of the client.</td>
</tr>
<tr>
<td>Database roles not working as expected.</td>
<td>If you are using database roles, then the Reports Server gets and then sets the default roles for the job request’s database connection. If the default roles require a password, then the Reports Server will log off and then back on to the database. As a result, it is best to include roles that require passwords in the report itself using the Role Name report property. Since the Reports Server gets and then sets the default roles on a per job basis, you cannot share roles between jobs. This is done to preserve security.</td>
</tr>
<tr>
<td>Problem Description</td>
<td>Probable Cause and Solution</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>URL mapping is not working.</td>
<td>Be sure you have a valid key mapping file. It must either be named cgicmd.dat (for the Reports Web CGI or Servlet) or owscmd.dat (for the Reports Web Cartridge) in the REPORT60 directory, or named according to the value set in the REPORTS60_CGIMAP or REPORTS60_OWSMAP environment variable.</td>
</tr>
<tr>
<td></td>
<td>To ensure the key mapping file can be found, first try (the following is a Web CGI example):</td>
</tr>
<tr>
<td></td>
<td>http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe/showmap?</td>
</tr>
<tr>
<td></td>
<td>and verify that your key entry has been correctly parsed in the resulting page. Then try, running the report using the key map entry:</td>
</tr>
<tr>
<td></td>
<td>http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe?your_key</td>
</tr>
<tr>
<td></td>
<td>where your_key is a valid key entry in the key mapping file.</td>
</tr>
<tr>
<td>Cannot shutdown the queue from the Reports</td>
<td>You should not leave the username and password blank the first time that you log in as the administrator. The first time that you log in as the queue administrator from the Reports Queue Manager (Options → Privileges → Administrator), you can specify any username and password. The username and password that you specify the first time will be the administrator's until you change it.</td>
</tr>
<tr>
<td>Queue Manager.</td>
<td></td>
</tr>
<tr>
<td>Cannot run the Reports Server as an NT Service</td>
<td>If the Reports Server is to be run as an NT service under the LocalSystem userid, then the system administrator must ensure that the following line is in the sqlnet.ora file, otherwise the server cannot be accessed:</td>
</tr>
<tr>
<td>under LocalSystem.</td>
<td>sqlnet.authentication_services=(NONE)</td>
</tr>
<tr>
<td>Problems finding files.</td>
<td>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 path names (for example, \SALES\DOCUMENTS\REPORTS). This applies to Reports Server parameters, Web CGI, Servlet, and Web Cartridge command mappings, and each hard-coded path name in each report being run.</td>
</tr>
</tbody>
</table>
The Web server reports an error opening the report output. If the Web server reports an error opening the report output, then check the name and extension carefully. On UNIX machines, the actual report name must have the same case as specified in the URL. If you are on Windows using the Windows Explorer, then be sure not to hide extensions for displayed files (View → Options) in the Explorer window that you are copying and renaming. This prevents you from creating files with names like your_report.rdf.txt. Alternatively, use a DOS window for file manipulation.

Report runs fine on design platform (for example, Windows), but fails on server platform (for example, UNIX). Check whether the release you are using on the design platform is the same as that on the server. If they are not the same, it could be that a difference between the two releases is causing the problem.

An invalid package was created when trying to create access to an Reports Developer report definition file in WebDB. In WebDB, verify the access controls that you defined for the printer, Reports Server, and report definition file. Check for the following:

- The OS Printer name defined in the Printer Access wizard is correct. If the printer does not appear in the Required Parameters page of the Report Definition File Access wizard, it is possible that you incorrectly entered the OS Printer name.

- Access to the Reports Server and optionally, the printer has been created.

- Users who require access to the report definition files, servers, and printer have been given access to them.

Make the necessary changes and then try to create a valid production package for the report definition file. You must create a valid production package in order to run this restricted report to a restricted Reports Server.

**Table F–1 Troubleshooting information**

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Probable Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Web server reports an error opening the report output.</td>
<td>If the Web server reports an error opening the report output, then check the name and extension carefully. On UNIX machines, the actual report name must have the same case as specified in the URL. If you are on Windows using the Windows Explorer, then be sure not to hide extensions for displayed files (View → Options) in the Explorer window that you are copying and renaming. This prevents you from creating files with names like your_report.rdf.txt. Alternatively, use a DOS window for file manipulation.</td>
</tr>
<tr>
<td>Report runs fine on design platform (for example, Windows), but fails on server platform (for example, UNIX).</td>
<td>Check whether the release you are using on the design platform is the same as that on the server. If they are not the same, it could be that a difference between the two releases is causing the problem.</td>
</tr>
</tbody>
</table>
| An invalid package was created when trying to create access to an Reports Developer report definition file in WebDB. | In WebDB, verify the access controls that you defined for the printer, Reports Server, and report definition file. Check for the following:

- The OS Printer name defined in the Printer Access wizard is correct. If the printer does not appear in the Required Parameters page of the Report Definition File Access wizard, it is possible that you incorrectly entered the OS Printer name.

- Access to the Reports Server and optionally, the printer has been created.

- Users who require access to the report definition files, servers, and printer have been given access to them.

Make the necessary changes and then try to create a valid production package for the report definition file. You must create a valid production package in order to run this restricted report to a restricted Reports Server. |
Table F–1  Troubleshooting information

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Probable Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports are not running when the URL is requested.</td>
<td>■ Make sure the Web server is responding (for example, by trying to bring up your Web server administration page). Refer to your Web server installation documentation.</td>
</tr>
</tbody>
</table>
| | ■ Make sure your Web CGI, Servlet, or Web Cartridge executable has been found and is responding. For Windows 95/NT, type one of the following in your browser URL field:  
http://your_webserver/your_virtual.cgi_dir/rwcgim60.exe or http://your_webserver/rwows  
For UNIX, type:  
http://your_webserver/your_virtual.cgi_dir/rw CGI60 or http://your_webserver/rwows  
A help page should appear. If it does not, check the mapping of your_virtual.cgi_dir (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 RWCGI60 executable in this physical directory. |
| | ■ Make sure that the REPORTS60_CGINODIAG (for Web CGI or Servlet) or REPORTS60_OWSNODIAG (for Web Cartridge) environment variable is not defined, otherwise all diagnostic output will be disabled. Test this by typing one of the following:  
http://your_webserver/your_virtual.cgi_dir/rw CGI60.exe/showenv?  
http://your_webserver/rwows/showenv?  
This will also allow you to view the other cartridge parameters or environment variables. |
Be sure the REPORTS60_PATH environment variable is defined. Check the environment variable by typing one of the following:

```
http://your_webserver/you_virtual_cgi_dir/rwcgi60.exe/showenv
http://your_webserver/rwows/showenv
```

Try running a simple report to your browser, by typing one of the following:

```
http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe?server=your_repserver+report=your_report.rdf+userid=scott/tiger@mydb+desformat=html
http://your_webserver/rwows?server=your_repserver+report=your_report.rdf+userid=scott/tiger@mydb+desformat=html
```

If the report does not display, then check to be sure that:

- Your_report.rdf runs correctly from Report Builder or Reports Runtime. Your_report.rdf is located in a directory specified under REPORTS60_PATH.
- The database connection string is correct.
- The Reports Server you are trying to run your report to may be restricted. If so, then you will need to be given access privileges to the server. Contact your Reports Developer system administrator.
- The report you are trying to run may be restricted. If so, then you will need to be given access privileges to run it to a restricted Reports Server. Contact your Reports Developer system administrator.

Remember that the Reports Server must have access to the report and any external files used by the report.

When sending a report to the Reports Server, you should only use the In Report value for parameters if they have their values explicitly set in the report definition. For example, suppose that you are launching a report from the Reports Queue Manager (Job → New). If you specify In Report for the Report Mode and Orientation parameters, and neither of them has a value specified in the report definition, then the job will fail.

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Probable Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report does not output to the printer</td>
<td>You may have access privileges to run a report to restricted Reports Server, but may not have access privileges to the printer you are trying to output to. Contact the Reports Developer system administrator.</td>
</tr>
</tbody>
</table>
Host name lookup failure

You typed an incorrect URL when trying to run a report request. Resubmit the report request using the correct URL. If you are unsure of the URL, contact your system administrator.

If you trying to run your report to a restricted Reports Server, the Web Gateway URL defined in the Server Access in WebDB maybe incorrect.


Note: Only users with Reports Developer system administrator privileges can access Oracle Reports Developer Security in WebDB.

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Probable Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name lookup failure</td>
<td>You typed an incorrect URL when trying to run a report request. Resubmit the report request using the correct URL. If you are unsure of the URL, contact your system administrator. If you trying to run your report to a restricted Reports Server, the Web Gateway URL defined in the Server Access in WebDB maybe incorrect. In WebDB, click Administrator from the WebDB main menu. Then, click Oracle Reports Developer Security and Server Access. Search for the Reports Server Access you want to edit. Confirm the Web Gateway URL on the Server Name and Printers page of the Server Access wizard. Note: Only users with Reports Developer system administrator privileges can access Oracle Reports Developer Security in WebDB.</td>
</tr>
</tbody>
</table>
Authentication
The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to allowing access to resources in a system.

Cache
A temporary storage place for database data that is currently being accessed or changed by users, or for data that Oracle Server requires to support users. The terms are often used interchangeably.

Cartridge
A manageable object that plugs in to the client, application server, or database through an open API.

CGI (Common Gateway Interface)
The industry-standard technique for running applications on a Web server. CGI enables a program running on the Web server to communicate with another computer to dynamically generate HTML documents in response to user-entered information.

Cookie
A cookie is a special text file that a Web server puts on the users hard disk so that it can remember something about the user at a later time. When users run report requests to a secured Reports Server, they must authenticate. If they successfully log on, their browser is sent an encrypted cookie. When a cookie has expired, subsequent requests (that is, ones that sent to a secured Reports Server) must re-authenticate.
CSS (Cascading Style Sheets)
HTML with CSS allows developers to control the style and layout of multiple Web pages all at once. A style sheet works like a template, a collection of style information, such as font attributes and color. Cascading refers to a set of rules that Web browsers use to determine how to use the style information. Navigator 4.0 and Internet Explorer 4.0 support cascading style sheets.

Domain
A grouping of network objects, such as databases, that simplifies the naming of network services.

Fail-over
The ability to reconfigure a computing system to utilize an alternate active component when a similar component fails.

HTML (Hypertext Markup Language)
A tag-based ASCII language used to specify the content and 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 Web traffic between a Web browser computer and the Web server being accessed.

IP (Internet Protocol)
The basic protocol of the Internet. It enables the delivery of individual packets from one host to another. It makes no guarantees about whether or not the packet will be delivered, how long it will take, or if multiple packets will arrive in the order they were sent. Protocols built on top of this add the notions of connection and reliability.

Net8
Oracle’s remote data access software that enables both client-server and server-server communications across any network. Net8 supports distributed processing and distributed database capability and runs over and interconnects many communication protocols.
OAS (Oracle Application Server)
Oracle Application Server is a strategic platform for network application deployment. By moving application logic to application servers and deploying network clients, organizations can realize substantial savings through reduced complexity, better manageability, and simplified development and deployment. Oracle Application Server provides the only business-critical platform that offers easy database web publishing and complete legacy integration while transition from traditional client-server to network application architectures.

ORACLE_HOME
An alternate name for the top directory in the Oracle directory hierarchy on some directory-based operating systems. 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.

Placeholder username
A placeholder username enables users to log on to the database using their personal username rather than the Oracle database username (for example, $user_name@database). A placeholder username allows:

- Users to log on only once to run multiple reports from the same database.
- Multiple end users to run the same report with personalized results (for example, one user may receive East coast sales results and another may receive West coast sales results).

The first time users log on to the database, however, they must log on using the Oracle username and password. For subsequent requests, the Reports Server looks for the user’s personal username in the database connection table. If it is found, the Reports Server gets the corresponding password from the cookie and runs the report.

Port
A number that TCP uses to route transmitted data to and from a particular program.
Push delivery
The delivery of information on the Web that is initiated by the server rather than by a client request. The Reports Server can push reports to WebDB site by scheduling the report request to run automatically on a secured Reports Server. The end user clicks the link on the WebDB site to view the report.

Reports Queue Manager
Enables you to monitor and manipulate job requests that have been sent to the Reports Server.

Reports Launcher
An application that utilizes the functionality provided by the Reports ActiveX control, such submitting a request to run the specified report to the Reports Server.

Reports Server
Enables you to run reports on a remote server in a multi-tier architecture. It can be installed on Windows NT, Windows 95, or UNIX. The Reports Server handles client requests to run reports by entering all requests into a job queue.

Reports Servlet
An interface between a Java-based Web server and Reports Runtime, enabling you to run report dynamically from your Web browser.

Reports Web CGI
An interface between a CGI-aware Web server and Reports Runtime, enabling you to run a report dynamically from your Web browser.

RWCLI60
An executable that parses and transfers the command line to the specified Reports Server (RWMTS60).

TCP/IP (Transmission Control Protocol based on Internet Protocol)
An Internet protocol that provides for the reliable delivery of streams of data from one host to another.

Tnsnames.ora
An Net8 file that contains connect descriptions mapped to service names. The file may be maintained centrally or locally, for use by all or individual clients.
**URI (Uniform Resource Identifier)**

A compact string representation of a location (URL) for use in identifying an abstract or physical resource. URI is one of many addressing schemes, or protocols, invented for the Internet for the purpose of accessing objects using an encoded address string.

**URL (Uniform Resource Locator)**

A URL, a form of URI, is a compact string representation of the location for a resource that is available via the Internet. It is also the text-string format clients use to encode requests to Oracle Application Server.

**Web browser**

A program that end users utilize to read HTML documents and programs stored on a computer (serviced by a Web server).

**WebDB**

Oracle WebDB is an HTML-based development tool for building scalable, secure, extensible HTML applications and Web sites. Oracle Reports Developer uses WebDB to control end user access to reports published on the Web by storing information about report requests, the secured server, and any Reports Developer printer used to print report output.

**WebDB component**

A PL/SQL stored procedure created by a WebDB component wizard (for example, a chart, form, or Reports Developer report definition file package). Executing the stored procedure creates the HTML code used to display the component.

**Web Server**

A server process (HTTP daemon) running at a Web site which sends out Web pages in response to HTTP requests from remote Web browsers.
### Symbols

<table>
<thead>
<tr>
<th>Tag</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!-- --&gt;</td>
<td>115</td>
</tr>
<tr>
<td>&lt;![CDATA[]]&gt;</td>
<td>115</td>
</tr>
<tr>
<td>&lt;condition&gt;</td>
<td>116</td>
</tr>
<tr>
<td>&lt;customize&gt;</td>
<td>118</td>
</tr>
<tr>
<td>&lt;data&gt;</td>
<td>119</td>
</tr>
<tr>
<td>&lt;dataSource&gt;</td>
<td>120</td>
</tr>
<tr>
<td>&lt;exception&gt;</td>
<td>121</td>
</tr>
<tr>
<td>&lt;field&gt;</td>
<td>124</td>
</tr>
<tr>
<td>&lt;formLike&gt;</td>
<td>128</td>
</tr>
<tr>
<td>&lt;formula&gt;</td>
<td>129</td>
</tr>
<tr>
<td>&lt;function&gt;</td>
<td>131</td>
</tr>
<tr>
<td>&lt;group&gt;</td>
<td>133</td>
</tr>
<tr>
<td>&lt;groupAbove&gt;</td>
<td>134</td>
</tr>
<tr>
<td>&lt;groupLeft&gt;</td>
<td>135</td>
</tr>
<tr>
<td>&lt;labelAttribute&gt;</td>
<td>136</td>
</tr>
<tr>
<td>&lt;layout&gt;</td>
<td>138</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
<td>141</td>
</tr>
<tr>
<td>&lt;matrix&gt;</td>
<td>143</td>
</tr>
<tr>
<td>&lt;matrixCell&gt;</td>
<td>146</td>
</tr>
<tr>
<td>&lt;matrixCol&gt;</td>
<td>146</td>
</tr>
<tr>
<td>&lt;matrixRow&gt;</td>
<td>147</td>
</tr>
<tr>
<td>&lt;object&gt;</td>
<td>148</td>
</tr>
<tr>
<td>&lt;programUnits&gt;</td>
<td>150</td>
</tr>
<tr>
<td>&lt;properties&gt;</td>
<td>151</td>
</tr>
<tr>
<td>&lt;property&gt;</td>
<td>152</td>
</tr>
<tr>
<td>&lt;report&gt;</td>
<td>155</td>
</tr>
<tr>
<td>&lt;section&gt;</td>
<td>157</td>
</tr>
<tr>
<td>&lt;select&gt;</td>
<td>158</td>
</tr>
<tr>
<td>&lt;summary&gt;</td>
<td>159</td>
</tr>
<tr>
<td>&lt;tabular&gt;</td>
<td>164</td>
</tr>
</tbody>
</table>

### A

- **access control**
  - availability calendars, 51
  - batch registering reports, 43
  - debugging, 64
  - example, 45
  - installing security packages in WebDB, 44
  - Oracle Reports system administrator, 50
  - overview, 42
  - printers, 56
  - report definition files, 58
  - setting default parameters, 66
- **accessing**
  - database, 46
  - demo tables, 46
- **ActiveX request method**, 34
- **architecture**
  - Reports Server, 4
  - Reports Server tier
    - database, 4
    - Reports Server, 4
    - Web server, 4
  - Reports Server tiers
    - thin client, 4
  - Web
    - server configurations, 5
- **authentication cookie**
  - expiring, 200
  - setting domain, 45
- **automatic installation for Reports Server**, 12
- **availability calendars**, 51
B
BATCH, 26
batch
    registering, 43
    reporting
        from a WebDB site, 70
    Reports Queue Manager, 39
batch modifications to reports, 109
batch reporting
    from a WebDB site, 39

C
    cache size, 203
    CGI, 11
    clustering
        configuration, 77
        resubmitting, 81
        running reports, 81
    command line arguments, 167
    commands
        RWMTS60, 22
    concepts, 3
    configuration
        Reports Server
            file, 22
        Reports Server overview, 15
        Web CGI example, 15
    customization
        overview, 86
        XML report definition, 89

D
    Data Access Descriptor See WebDB
    Listener
database tier
    Reports Server, 4
debugging
    restricted reports, 64
    tracing options, 111
    XML report definitions, 110
default printer
    set access, 30
    setting access, 28
demo table, accessing, 46
duplicate job detection
    multiple output destinations, 35
    Reports Server handling, 34, 194

E
evironment variables
    UNIX
        REPORTS60_PATH, 21
examples
    <!-- -->, 115
    <![CDATA[ ]]>), 115
    <condition>, 118
    <customize>, 118
    <data>, 119
    <dataSource>, 120
    <exception>, 123
    <field>, 127
    <formLike>, 130
    <formula>, 129
    <function>, 132
    <group>, 133
    <groupAbove>, 134
    <groupLeft>, 135
    <labelAttribute>, 138
    <layout>, 138
    <link>, 142
    <matrix>, 144
    <matrixCell>, 146
    <matrixCol>, 147
    <matrixRow>, 148
    <object>, 149
    <programUnits>, 150
    <properties>, 151
    <property>, 154
    <report>, 155
    <section>, 158
    <select>, 158
    <summary>, 163
    <tabular>, 164
    access control, 45
    cluster configuration, 77
    full URL syntax, 38
    key mapping, 36, 37
RWCLI60 command line request, 33
simplified URL syntax, 39
Web CGI configuration, 15
XML report definitions
  additional objects, 96
  formatting, 92
  formatting exception, 94
  full, 98
  hyperlink, 95
  PL/SQL, 95

I
installation
  additional Reports Server, 26
  automatic, 12
  manual, 15, 16
  Reports Server Security, 43
  starting as Windows NT service, 28
  starting non-service in Windows NT, 31
  starting Reports Server on UNIX, 29, 30
  using Smart Server Install, 11
invalid package procedure, 63

K
key mapping
  cgicmd.dat (Web CGI), 36
  enabling, 36
  example, 36, 37
  mapping entries, 37
  when to use, 35

L
load balancing
  configuration, 77
  resubmitting jobs, 81
  running reports, 81

M
manual configuration for Reports Server, 15

N
new features, xiii

P
parameters
  Reports Server configuration, 203
  RWCLI60 command line arguments, 167
  SOURCEDIR, 20
processing
  Web reports, 6

Q
queue activity
  database, 24

R
registry entries, 209
report request methods
  ActiveX, 34
  RWCLI60 command line, 33
  SRW.RUN_REPORT, 34
  URL syntax, 34
report requests
  building reports, 37
  duplicate job detection, 34
  from Manage Component page, 63
  in WebDB site, 67
  running from a browser, 38
  scheduling, 39
  scheduling in WebDB, 70
  specifying request, 37
  when servers are clustered, 81
report source path setting, 29
Reports Queue Manager
  monitoring job status, 23
  scheduling jobs to run, 39
parser, 111
partial, 90
PL/SQL example, 95
required tags, 89
running, 109
running to Report Builder, 113
tags, 114
writing to files, 114