Oracle® Reports Services

Publishing Reports to the Web with Oracle Internet Application Server

for Windows NT and UNIX

Release 1.0.1

Part No. A83592-02

July 2000

ORACLE®

Publishing Reports to the Web with Oracle Internet Application Server, Release 1.0.1

Part No. A83592-02

Contents

Part I   Publishing Reports

1   Publishing Architecture and Concepts

2 Installing Oracle Internet Application Server with Oracle Reports Services

3 Configuring the Oracle Reports Server on Windows NT and UNIX

4   Running Report Requests

5   Controlling User Access to Reports

6   Configuring Oracle Reports Services Clusters

7 Customizing Reports at Runtime

Part II   Appendixes

A   RWCLI60 Command Line Arguments

B Oracle Reports Services Configuration Parameters

C Environment Variables

D Database Connection Strings

E Migrating from Web Cartridge to Web CGI

F Troubleshooting

Glossary

Index

x

Send Us Your Comments

Publishing Reports to the Web with Oracle Internet Application Server, Release 1.0.1

Part No. A83592-02

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, then 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, then please indicate the chapter, section, and page number (if available). You can send comments to us at:

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

Preface

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

Oracle Reports Services New Features

New Feature	See
Oracle Internet Applications Server. Provides a middle-tier application server.	Chapter 2, "Installing Oracle Internet Application Server with Oracle Reports Services"
Oracle HTTP Server *powered by Apache* for JSDK. Configuration for running the Oracle Reports Server Servlet with JSDK through the Oracle HTTP Server *powered by Apache*.	Section 3.3.1, "Configuring the Oracle Reports Servlet with JSDK"
Oracle HTTP Server *powered by Apache* for JServ. Configuration for running the Oracle Reports Server Servlet through the Oracle HTTP Server *powered by Apache*.	Section 3.3.2, "Configuring the Oracle Reports Servlet with JServ"
Oracle HTTP Server *powered by Apache* Listener. Configuration for Oracle HTTP Server *powered by Apache* Listener to run the Oracle Reports Server CGI.	Section 3.4, "Configuring the Oracle HTTP Server powered by Apache Listener"
Control user access to report. Restrict user access to reports that are run with Oracle Reports Services. Oracle Reports Services uses Oracle WebDB to check that users have the necessary access privileges to run the report with restricted Oracle Reports Services.	Chapter 5, "Controlling User Access to Reports"
Clustering. Improve performance and loading balancing by clustering your Oracle Reports Services.	Chapter 6, "Configuring Oracle Reports Services Clusters"

New Feature	See
Report Customization. Apply customizations to reports at runtime without changing the original report and generate different output depending upon the audience.	Chapter 7, "Customizing Reports at Runtime"

Intended Audience

This manual is intended for anyone who is interested in publishing reports with Oracle Reports Services. Perhaps you have built reports yourself and now want to publish them to a wider audience in your organization. Or perhaps someone else built the reports for you and you now want to deploy them for other users to access. To configure Oracle Reports Services 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, then you should also be knowledgeable about your Web server configuration.

Structure

This manual contains the following chapters:

Chapter 1	Introduces the architecture of the Oracle Reports Services and choices that you need to make before you configure the report.
Chapter 2	Provides information about installing.
Chapter 3	Describes how to configure the Oracle Reports Server.
Chapter 4	Describes the various methods for running reports to the Oracle Reports Services.
Chapter 5	Describes how the Oracle Reports Services can be integrated with Oracle WebDB to control user access to reports.
Chapter 6	Describes how to configure the Oracle Reports Services 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 Services, 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 Internet Application Server*, A83591-01

Notational Conventions

The following conventions are used in this book:

Convention	Meaning
boldface text	Used for emphasis. Also used for menu items, button names, labels, and other user interface elements.
italicized text	Used to introduce new terms.
courier font	Used for path and file names, and for code and text that you type.
COURIER CAPS	Used for file extensions (.PLL or .FMX) and SQL commands
CAPS	Used for environment variables, built-ins and package names, and executable names

Part I

Publishing Reports

1

Publishing Architecture and Concepts

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 Services provides an unbounded, easy-to-use, scalable, and manageable solution for high-quality database publishing and reporting.

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

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

1.1 Oracle Reports Services

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

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

Oracle Reports Services 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 Oracle Reports Queue Manager (RWRQM60) or through the API. The Oracle Reports Queue Manager can reside on the same machine as Oracle Reports Services or on a client machine. On UNIX, you can use the Oracle Reports Queue Viewer (RWRQV60) to view the Oracle Reports Services queue.

1.2 Oracle Reports Services Architecture

Oracle Reports Services can be configured in a number of ways depending upon your requirements. When used in a Web environment, the Oracle Reports Services architecture consists of four tiers[1]:

- The thin client tier

- The Web server tier

- The Oracle Reports Services 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.

1.2.1 Web Architecture: Server Configurations

The diagrams that follow illustrate two of the most common configurations for Oracle Reports Services in a Web environment. The key difference between the two configurations is whether Oracle Reports Services and Web server tiers are on the same or different machines. In the first case, the Web server and Oracle Reports Services 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.

---

[1] The term tier refers to the logical location of the components that comprise the Oracle Reports Services architecture. Each of the tiers, though, could reside on the same or different machines.

Figure 1–1   Web Architecture, Three Machine Configuration
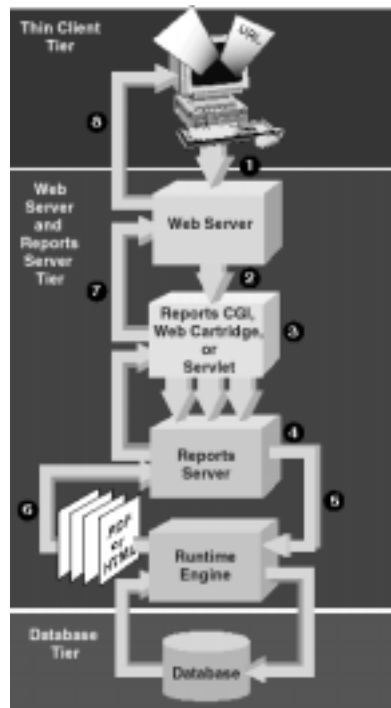
Figure 1–2   Web Architecture, Four Machine Configuration
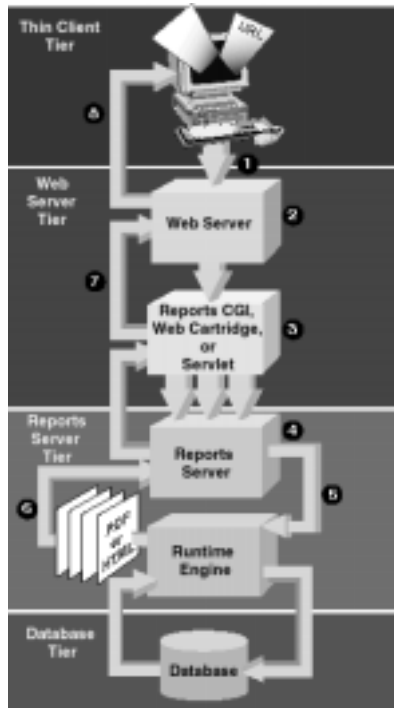
1.2.1.1 Processing Web Reports

Web reports are processed as follows:

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 Oracle Reports Server Web CGI or Oracle Reports Server Servlet, depending upon which one you have configured.

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

4. If the request includes a time tolerance[1], then Oracle Reports Services checks its output cache to determine whether it already has output that satisfies the request. If it finds acceptable output in its cache, then it will immediately return that output rather than executing the report.

5. Oracle Reports Services 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 runs the report.

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

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

1.2.2 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 Oracle Reports Services from a thin client such as the Oracle Reports Launcher or command line, RWCLI60. 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 Oracle Reports Services, you can include a TOLERANCE argument. TOLERANCE defines the oldest output that the requester would consider acceptable. For example, if the requester specified five minutes as the TOLERANCE, Oracle Reports Services would check its cache for duplicate report output that had been generated within the last five minutes.

[2] When you configure Oracle Reports Services, you can specify the maximum number of runtime engines it can use. If Oracle Reports Services is under this maximum, then it can start new runtime engines to handle requests. Otherwise, the request must wait until one of the current runtime engines completes its current job.

Figure 1–3   Non-Web Architecture
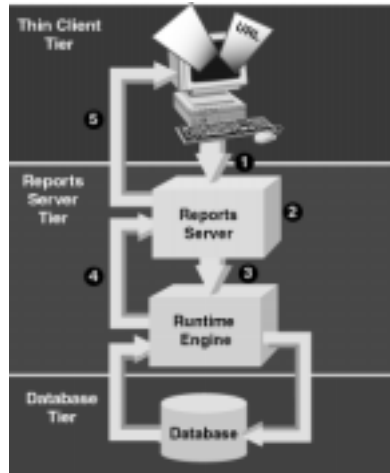
1.2.2.1 Processing Reports

In a non-Web environment, reports are processed as follows:

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

2. Oracle Reports Services 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 runs the report.

4. Oracle Reports Services is notified that the job has been completed.

5. If Oracle Reports Services was called synchronously, then it signals the client that the job has been completed. If the destination type (DESTYPE) for the command line client is set to localfile in the job request, then the output is transferred to the client.

1.3 Oracle Reports Services Configuration Choices

The configuration of Oracle Reports Services can vary widely depending upon the requirements of your system. Before attempting to configure Oracle Reports Services, 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.

1.3.1 Enable Web and Non-Web Requests

As you saw in Section 1.2, "Oracle Reports Services Architecture", Oracle Reports Services 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 Oracle Reports Thin Client, which is comprised of the Oracle Reports Launcher, the Oracle Reports Queue Manager, and RWCLI60).

To enable users to launch reports from a Web client, you need to install either the Oracle Reports Services Web CGI or Servlet with your Web server to communicate between the Web server and Oracle Reports Services. 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 1.3.2, "Choose the Oracle Reports Services 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 Oracle Reports Services 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 Oracle Reports Services 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 Oracle Reports Services.

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

1.3.2 Choose the Oracle Reports Services Web CGI or Servlet

As discussed in Section 1.3.1, "Enable Web and Non-Web Requests", to use Oracle
Reports Services in a Web environment, you must install and configure the Oracle
Reports Services Web CGI or Servlet to handle the transmission of job requests and
output between your Web server and Oracle Reports Services. The key
consideration in this choice is the following:

- If you are using a CGI-aware Web server (for example, Oracle Internet
  Application Server or WebDB Listener), then choose the Oracle Reports Services
  Web CGI.

- If you are using a Java-based Web server, then choose the Oracle Reports
  Services Servlet.

1.3.3 Choose the Location of Oracle Reports Services

As described in the Section 1.2, "Oracle Reports Services Architecture", you can
place Oracle Reports Services on the same machine as your Web server or on a
different machine. As you make this decision, you should consider the following:

- Having Oracle Reports Services 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, then you need to take that into account when determining
  the machine's resource requirements (that is, memory and disk space).

- Having Oracle Reports Developer and the Web server on the same machine
  reduces network traffic. The Oracle Reports Services CGI or Servlet must reside
  on the same machine as the Web server. If Oracle Reports Services is on a
  different machine, then its transmissions to the Oracle Reports Services CGI and
  Servlet must travel across a network. If it is on the same machine, then the
  transmissions do not have to travel across the network.

Chapter 3, "Configuring the Oracle Reports Server on Windows NT and UNIX"
provides guidelines for configuring Oracle Reports Service using the Oracle Reports
Services Servlet.

2

Installing Oracle Internet Application Server with Oracle Reports Services

The Oracle Reports Services is installed as part of the Enterprise Edition of Oracle Internet Application Server (*i*AS). The Enterprise Edition is recommended for medium to large sized Web sites that handle a high volume of transactions.

For your convenience, the Oracle HTTP Server *powered by Apache*, a Web listener that supports the Common Gateway Interface (CGI), is provided. The Oracle HTTP Server *powered by Apache* can be installed through the Oracle Universal Installer, which is provided with the Oracle Internet Application Server.

For more detailed information about installing Oracle Reports Services, refer to the *Oracle Internet Application Server Installation Guide.* All necessary requirements and tasks are documented in this manual.

2.1 About the Oracle Universal Installer

The Oracle Internet Application Server uses the Oracle Universal Installer, a Java-based tool to configure environment variables and to install components. The installer guides you through each step of the installation process, so you can choose different configuration options.

The installer includes features that perform the following tasks:

- Explore and provide installation options for the product.

- Detect preset environment variables and configuration settings.

- Set environment variables and configuration settings during installation.

- Deinstall the product.

2.2 About the Oracle HTTP Server *powered by Apache*

The Oracle Internet Application Server uses the Oracle HTTP Server *powered by Apache* Web server technology. Using the Apache Web server technology offers the following:

- Scalability

- Stability

- Speed

- Extensibility

The Apache server delegates the handling of HTTP requests to its modules (mods), which add functionality not included in the server by default. Using the Apache APIs, it is easy to extend the Apache functionality. A large number of mods have already been created and are included on your CD-ROM. Although the default Apache HTTP server supports only stateless transactions,[1] you can configure it to support stateful transactions[2] by leveraging the functionality supplied by Apache JServ (mod_jserv), which is described in the *Oracle Internet Applications Server 8i Overview Guide*.

---

[1] A stateless transaction consists of a request and a response. In a stateless transaction, no information about the user (the requestor) is tracked by the system, and each transaction is unrelated to those that precede or follow it.

[2] Stateful transactions are similar to database sessions because information abut the user (the initiator of the transaction) is tracked by the system for one or more phases of the transaction. In addition to user information, with a stateful transaction, the system also keeps track of the state (the set of conditions at a moment in time) of one or more preceding events in the sequence of a transaction.

Additional information about the Oracle HTTP Server *powered by Apache* can also be found in the *Oracle Internet Application Server Installation Guide* and the *Apache Web Server, Release 1.3.9* manual on your CD-ROM.

3

Configuring the Oracle Reports Server on Windows NT and UNIX

When you install the Oracle Internet Application Server (*i*AS) with the Oracle HTTP Server *powered by Apache*, the Oracle Reports Server Servlet and the Oracle Reports Server CGI are automatically configured for you in the Windows NT and UNIX environments. This chapter describes how to manually change the configurations that were provided by default.

This chapter also describes how to start and stop the Oracle Reports Server and the configuration environment variables.

3.1 Starting and Stopping the Oracle Reports Server

Throughout this chapter you are asked to start, stop, and restart the Oracle Reports Server. Following are the instructions for doing this.

3.1.1 Starting the Oracle Reports Server

The following sections describe how to start the Oracle Reports Server on Windows NT or on UNIX.

3.1.1.1 Starting the Oracle Reports Server on Windows NT

Proceed with the following steps to start the Oracle Reports Server on Windows NT:

1. On the Oracle 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]** (where `repserver` is the name of the Oracle Reports Server instance) 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.

   If you want to output to PostScript or to a printer, then ensure the user running the Oracle 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 Oracle Reports Server has started. If your Oracle Reports Server cannot start, then refer to Appendix F, "Troubleshooting" for more information.

   When you start the Oracle Reports Server for the first time, an Oracle Reports Server configuration file (for example, `repserver.ora`) is created in the `ORACLE_HOME\REPORT60\SERVER` directory. The setting for your Oracle 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 Oracle Reports Server for the changes to take effect.

3.1.1.2 Starting the Oracle Reports Server on UNIX

Do the following steps to start the Oracle Reports Server on UNIX:

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

```
rwmts60 name=repserver
```

Run the following command to run the Oracle Reports Server in the background:

```
rwmts60 name=repserver &
```

2. From the `$ORACLE_HOME/BIN` directory, run the following command line to ensure the Oracle Reports Server is running:

```
rwrqv60 server=repserver
```

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

If you want to output to PostScript or to a printer, then the printer must be configured in the `uiprint.txt` file (this file is located in the `$ORACLE_HOME/guicommon6/tk60/ADMIN` directory).

3.1.2 Starting the Oracle Reports Server on Windows NT as a Non-Service

Run the following command:

```
rwmts60 -listen repserver
```

Or in batch mode:

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

The `repserver` does not need to have the domain qualifier (for example, `.world`) appended to it.

3.2 Stopping the Oracle Reports Server

The following sections discuss how to stop the Oracle Reports Server on Windows NT and UNIX.

3.2.1 Stopping or Deinstalling the Oracle Reports Server on Windows NT

To stop the Oracle Reports Server on Windows NT, you do the following:

1. On the Oracle 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
   ```

   The `repserver` does not need to have the domain qualifier (for example, `.world`) appended to it.

3.2.2 Stopping the Oracle Reports Server on UNIX

Do one of the following to stop the Oracle Reports Server:

- If the Oracle Report Server is running in the foreground, then ensure that the focus is in the correct window and press **ctrl**-**C**.

- If the Oracle Report Server is running in the background, then enter the following at the command line:

  ```
  ps -ef |grep 'rwmts60'
  ```

  You would then enter:

  ```
  kill -9 process_number
  ```

3.3 Configuring the Oracle Reports Servlet

With the Oracle HTTP Server *powered by Apache*, there are two Oracle Reports Servlet configurations that you can manually change:

- Oracle Reports Server Servlet with JSDK

- Oracle Reports Server Servlet with JServ

3.3.1 Configuring the Oracle Reports Servlet with JSDK

The following configuration assumes that the Oracle HTTP Server *powered by Apach*e is installed in the following directory:

`/private1/ias`

It also assumes that the Oracle Reports Server is installed in the following directory:

`/private1/ias/6iserver`

You do the following steps to configure the Oracle Reports Servlet with JSDK:

1. Add the following entry to the Oracle Reports Servlet properties file, `servlet.properties`, (for example, the Oracle Reports Servlet properties file located in `/private1/ias/Apache/Jsdk/examples`):

   `servlet.RWServlet.code=oracle.reports.rwcgi.RWServlet`

2. Create the directory hierarchy `oracle/reports/rwcgi` in your Web server Java class directory:

   `/private1/ias/Apache/Jsdk/examples/oracle/reports/rwcgi`

   You then copy into this new directory the `RWServlet.class` file found in:

   `/private1/ias/6iserver/reports60/java`

3. Add the root directory from the previous step into your CLASSPATH environment variable, located in `/private1/ias/Apache/Ojsp`. Also add `Ojsp/lib/servlet.jar` to the CLASSPATH environment variable. For example:

   ```
   setenv CLASSPATH/private1/ias/Apache/jdk/bin:
   /private1/ias/Apache/jdk/lib/classes.zip:
   /private1/ias/Apache/Jsdk/examples:/private1/ias/Apache/Ojsp/lib/servlet.jar
   ```

4. Set the PATH variable by entering the following:

   ```
   setenv PATH /private1/ias/6iserver/bin:/private1/ias/Apache/Apache/bin:
   private1/ias/Apache/jdk/bin:
   private1/ias/Apache/jsdk/bin:$PATH
   ```

5. Start the Oracle Reports Server.

6. Start the Oracle Reports Servlet runner by running the following command:

   `servletrunner &`

7. Verify that the Oracle Reports Servlet is running by doing the following:

   a. Running the following from your browser to ensure the installation and setup are okay:

      ```
      http://hostname:portno/servlet/RWServlet/help?
      ```

      where:

`hostname`	is the machine name where the Apache listener is running.
`portno`	is the port number that where the Apache listener is started.

      This shows you that the Help page is active.

   b. Run the following from your browser to ensure the Oracle Reports Server is up:

      ```
      http://hostname:portno/servlet/RWServlet/showjobs?
      server=repserver
      ```

   c. Enter the following from your browser to run a report:

      ```
      http://hostname:portno/servlet/RWServlet?server=repserver+
      report=ReportName+destype=cache+userid=ConnectString+desformat=htmlcss
      ```

      You can also use the `cgicmd.dat` file for key mapping.

If you modify the configuration file, then you need to stop and restart the Oracle Reports Server to acknowledge the changes.

3.3.2 Configuring the Oracle Reports Servlet with JServ

You do the following to configure the Oracle HTTP Server *powered by Apache* to run the Oracle Reports Servlet with JServ. The changes are made in the ias_home/`Apache/Jserv/etc/jserv.properties` file, where ias_home is the location where you installed *i*AS:

1. Add the following line:

   ```
   wrapper.classpath=ias_home/Apache/Jserv/servlets
   ```

2. Change the `wrapper.env=ORACLE_HOME=ias_home` line to the following:

   ```
   wrapper.env=ORACLE_HOME=ias_home/6iserver
   ```

3. Change the `wrapper.env=LD_LIBRARY_PATH=ias_home/lib` line to the following:

```
wrapper.env=LD_LIBRARY_PATH=ias_home/lib:ias_home/6iserver/bin:ias_home/
6iserver/lib
```

4. Add the following line to the `Apache/Jserv/etc/zone.properties` file:

```
servlet..RWServlet.code=oracle.reports.rwcgi.RWServlet
```

5. Copy the `RWServlet.class` file to the following directory (you might need to create the directory):

```
ias_home/Apache/Jserv/servlets/oracle/reports/rwcgi
```

The `http://host:port/servlet/RWServlet` URL runs the servlet.

6. Start the Oracle Reports Server.

7. Start the Oracle HTTP Server *powered by Apache* listener using the following command:

```
httpdsctl start
```

8. Verify the Oracle Reports Server is running by:

    a. Run the following from your browser to ensure the installation and setup are okay:

```
http://hostname:portno/servlets/RWServlet/help?
```

    This shows you that the Help page is active.

    b. Run the following from your browser to ensure the Oracle Reports Server is up:

```
http://hostname:portno/servlets/RWServlet/showjobs?
server=repserver
```

    c. Enter the following from your browser to run a report:

```
http://hostname:portno/servlets/RWServlet?server=repserver+
report=ReportName+destype=cache+userid=ConnectString+desformat=htmlcss
```

    You can also use the `cgicmd.dat` file for key mapping.

If you modify the configuration file, then you need to stop and restart the Oracle Reports Server to acknowledge the changes.

3.4 Configuring the Oracle HTTP Server *powered by Apache* Listener

You do the following to change the default configuration for the Oracle HTTP Server *powered by Apache* listener to run the Oracle Reports CGI:

1. Add the following entry to the file `httpds.conf` (found in `/private1/ias/Apache/Apache/conf`):

    ```
    ScriptAlias /cgi-bin/        "/private1/ias/6iserver/bin"
    ```

2. Start the Oracle Reports Server.

3. Start the Oracle HTTP Server *powered by Apache* listener using the following command:

    ```
    httpdsctl start
    ```

4. Verify the Oracle Reports CGI is running by:

    a. Run the following from your browser to ensure the installation and setup are okay:

    ```
    http://hostname:portno/cgi-bin/rwcgi60/help?
    ```

    This shows you that the Help page is active.

    b. Run the following from your browser to ensure the Oracle Reports Server is up:

    ```
    http://hostname:portno/cgi-bin/rwcgi60/showjobs?
    server=repserver
    ```

    c. Enter the following from your browser to run a report:

    ```
    http://hostname:portno/cgi-bin/rwcgi60?server=repserver+
    report=ReportName+destype=cache+userid=ConnectString+desformat=htmlcss
    ```

    You can also use the `cgicmd.dat` file for key mapping.

If you modify the configuration file, then you need to stop and restart the Oracle Reports Server to acknowledge the changes.

3.5 Configuring the Web Server

In order to make this configuration example meaningful, it is necessary to make several assumptions:

- You are configuring the Oracle Reports Server to enable Web requests.
- You are using the Oracle Reports Server Web CGI with the CGI-aware Web server with Oracle HTTP Server *powered by Apache*.
- The Oracle Reports Server is installed on a different machine than the Web server.

The Web CGI-BIN directory on your Web server contains CGI executables. The following are performed on the Web server machine:

1. Start your Web server by entering the following:

2. Start your browser.

3. Create a listener.

4. Configure your Web server mapping and note the physical and virtual directories. For example:

Table 3–1   Web CGI-BIN Physical and Virtual Directories

Directory Description	Physical Directory example	Virtual Directory Example	Permissions Required
Web CGI-BIN	`c:\orant\oas\bin`	/CGI-BIN	execute
Apache Web server CGI-BIN	`c:\program files\Apache Group\Apache\cgi-bin`	/CGI-BIN	execute

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.

3.5.1 Configuring the Oracle Reports Web CGI

The following steps are performed on the Web server machine.

3.5.1.1 Configuring the Oracle Reports Web CGI

To configure the Oracle Reports Web CGI copy `rwcgi60.exe` (located in the `ORACLE_HOME\BIN` directory) to your CGI-BIN directory.

In Table 3–1, " Web CGI-BIN Physical and Virtual Directories" the Web CGI physical directory is `C:\your_webserver\bin`, or if you are using the Apache Web Server, `C:\Program Files\Apache Group\Apache\cgi-bin`.

The CGI-BIN directory is defined in your Web server configuration. The Oracle Reports Web CGI must be in a path mapped as a CGI directory. The Oracle Reports RDF files must be in a path only accessible to the Oracle Reports Server. If you choose the default installation of Oracle Reports Server 6*i* and the WebDB Listener, then you will find the `rwcgi60.exe` file in the following path:

```
D:\orant\bin\rwcgi.exe
```

3.5.1.2 Creating a Service Entry for the Oracle Reports Server

If the Web server is on a different machine than your Oracle Reports Server, then you must add the Oracle Reports Server service entry. This service entry was created on the Oracle Reports Server machine in 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 Oracle Reports Server.

If you do not remember the service entry settings for the Oracle Reports Server, then open the `tnsnames.ora` file located in the `ORACLE_HOME\NET80\ADMIN` directory on your Oracle 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 Oracle 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.

3.5.1.3 Setting the Default Oracle Reports Server (Optional)

You can, optionally, set defaults for the Oracle Reports Server on both the Windows NT platform or the UNIX platform.

3.5.1.3.1 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 the **Edit**→**New**→**String** value to add the following registry entry:

    REPORTS60_REPORTS_SERVER

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

3.5.1.3.2 **UNIX** For UNIX, set the REPORTS60_REPORTS_SERVER environment variable to the name of the Oracle Reports Server.

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, where repserver is the name of the Oracle Reports Server that you are configuring (the TNSnames service entry name of the Oracle Reports Server):

```
setenv REPORTS60_REPORTS_SERVER repserver
```

3.6 Configuring the Oracle Reports Server with Environment Variables

This section discusses how you can configure and start the Oracle Reports Server with environment variables

3.6.1 Configuring the Oracle Reports Server in Windows NT with Environment Variables

There are two primary steps for configuring the Oracle Reports Server in Windows NT with environment variables:

1. Setting the environment variables (optional)

2. Starting the Oracle Reports Server

3.6.1.1 Setting the Environment Variables (Optional)

You can set two optional environment variables. The first lets the Oracle 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.

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
```

Alternatively, after the Oracle Reports Server is installed, you can set the source path in the Oracle Reports Server configuration file. See the SOURCEDIR parameter in Appendix B, "Oracle Reports Services 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
```

3.6.1.2 Starting the Oracle Reports Server

To start the Oracle Reports Server, you do the following:

1. On the Oracle 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]** (where `repserver` is the name of the Oracle Reports Server instance) 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.

   If you want to output to PostScript or to a printer, then ensure the user running the Oracle 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 Oracle Reports Server has started. If your Oracle Reports Server cannot start, then refer to Appendix F, "Troubleshooting" for more information.

   When you start the Oracle Reports Server for the first time, an Oracle Reports Server configuration file (for example, `repserver.ora`) is created in the `ORACLE_HOME\REPORT60\SERVER` directory. The setting for your Oracle 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 Oracle Reports Server for the changes to take effect.

3.6.2 Configuring the Oracle Reports Server on UNIX with Environment Variables

There are two primary steps for configuring the Oracle Reports Server on UNIX with environment variables:

1. Setting environment variables (optional)

2. Starting the Oracle Reports Server

3.6.2.1 Setting the Environment Variables (Optional)

You can set two environment variables, REPORTS60_PATH and TNS_ADMIN. The REPORTS60_PATH is the search path for the Oracle Reports Server source files (for example, RDFs, TDFs, and PLLs), and TNS_ADMIN overrides the default location for `tnsnames.ora` and `sqlnet.ora`. To set these do the following:

1. Create a directory for your source reports (for example, `/WEB_REPORTS`).

2. Set the REPORTS60_PATH environment variable to locate the reports. For example, using the C shell syntax:

```
setenv REPORTS60_PATH /WEB_REPORTS
```

Alternatively, after the Oracle Reports Server is installed, you can set the source path by using the SOURCEDIR parameter. See Appendix B, "Oracle Reports Services Configuration Parameters" for more information.

3. Set the TNS_ADMIN environment variable to point to the location of the `tnsnames.ora` file. For example, using the C shell syntax:

```
setenv TNS_ADMIN $ORACLE_HOME/NET80/ADMIN
```

3.6.2.2 Starting the Oracle Reports Server on UNIX

Do the following steps to start the Oracle Reports Server on UNIX:

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

```
rwmts60 name=repserver
```

Run the following command to run the Oracle Reports Server in the background:

```
rwmts60 name=repserver &
```

2. From the `$ORACLE_HOME/BIN` directory, run the following command line to ensure the Oracle Reports Server is running:

```
rwrqv60 server=repserver
```

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

If you want to output to PostScript or to a printer, then the printer must be configured in the `uiprint.txt` file (this file is located in the `$ORACLE_HOME/guicommon6/tk60/ADMIN` directory).

3.7 Environment Variables

Environment variables are the configuration parameters that are used to control or customize the behavior of the Oracle Reports Server. Variables can be set using a command line for Windows NT and a shell script for UNIX.

Variable	Description
REPORTS60_COOKIE_EXPIRE	Determines the expire time of the cookie in minutes. The default value is 30.
	Cookies save encrypted user names and passwords on the client-side when users log on to a secured Oracle 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 a secured Oracle Reports Server), user must re-authenticate to run the report.
REPORTS60_DB_AUTH	Specifies the database authentication template used to log on to the database. The default value is `dbauth.htm`.
REPORTS60_ENCRYPTION_KEY	Specifies the encryption key used to encrypt the user name and password for the cookie. The encryption key can be any character string. The default value is reports6.0.

Variable	Description
REPORTS60_REPORTS_SERVER	Specifies the default Oracle Reports Server for Web 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.
REPORTS60_SSLPORT	If you are using SSL and you want to use a port number other than 443, then you can use this variable to set a different port number. The default value is 443.
REPORTS60_SYS_AUTH	Specifies the authentication template used to authenticate the user name and password when users run report request to a secured Oracle Reports Server. The default value is sysauth.htm.

3.8 Running a Report Request from a Web Browser

You do the following to run a report request from a Web browser:

1. Ensure the Oracle 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	is replaced with a valid database logon.
my_db	is replaced with tnsnames.ora entry you created for earlier for the Oracle Reports Server (Section 3.5.1.2, "Creating a Service Entry for the Oracle 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.

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

   2. View the status of the request (optional):

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

   - For UNIX run the following command:

     ```
     rwrqv60 server=repserver showjobs=current
     ```

3.8.1 Other Steps

You can also perform the following, additional, steps:

1. Tune the Oracle Reports Server (optional) to optimize performance or implement additional features, such as access control. Doing this step eliminates the need to show all of the parameters as shown in Section 3.8, "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.

3.9 Modifying the Oracle Reports Server Configuration (Optional)

When you start the Oracle Reports Server for the first time, the Oracle Reports Server is set with default configuration settings (for example, maximum and minimum engines). At some point, you might want to modify the Oracle 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 3.9.1, "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 Oracle Reports Server for load balancing, refer to Chapter 6, "Configuring Oracle Reports Services 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 Oracle Reports Server configuration file, see Appendix B, "Oracle Reports Services Configuration Parameters" for more information about the available configuration parameters.

- Modify environment variables, refer to Appendix C, "Environment Variables" for more information.

3.9.1 Updating the Database with Job Queue Activity

You can set up your database to take snapshots of the Oracle Reports Server queue activity whenever jobs are run. When you start the Oracle Reports Server, a connection to the database is made. By default, the Oracle 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 Oracle Reports Server). The prototype of the procedure (the procedure name and the parameters it expects) should not be edited.

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

3.9.1.1 On the Oracle Reports Server Machine

To update the database with job queue activity on the Oracle Reports Server machine, you do the following:

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

2. The `repserver_schema` must have, at a minimum, create table and create package privileges to run the `rw_server.sql` script 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>
```

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

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

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

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

> **Note:**   When you restart your Oracle Reports Server, queue activity in the database is deleted by default. You can override the default by editing the API.

4

Running Report Requests

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, described below:

- The RWCLI60 command line enables you to run a report request from the command line prompt. RWCLI60 is an executable file that parses and transfers the command line to the specified Oracle Reports Services. It uses a command line similar to the Oracle Reports Runtime executable file (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.

- The 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 Oracle Reports Services. 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.

- The WebDB component enables you to 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. Oracle Reports Services 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 Oracle Reports Services through industry-standard ActiveX technology enabling you to run reports from any ActiveX container. The Oracle Reports Launcher is an example of an ActiveX container. Refer to the ActiveX and Oracle Reports Launcher online help for more information.

- The SRW.RUN_REPORT is a packaged PL/SQL procedure that runs an Oracle Reports Runtime command. When you specify the SRW.RUN_REPORT command line, set the SERVER argument to Oracle Reports Services TNS service entry name to cause the SRW.RUN.REPORT command to behave as though you executed an RWCLI60 command. Refer to the Oracle 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 of minutes (that is, 0 or greater), a copy of the report output is saved in Oracle Reports Services'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. Oracle Reports Services reuses 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 Oracle Reports Queue Manager), then 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.

4.2.1 Usage Notes

You might find the following usage notes helpful:

- 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. Oracle Reports Services detects 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 Oracle Reports Queue Manager or manually by setting the CACHESIZE parameter in the Oracle Reports Services configuration file. Oracle Reports Services attempts 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 Oracle Reports Queue Manager.

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

- If a report is being processed when an identical job is submitted, then Oracle Reports Services reuses 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 Oracle Reports Services engines and someone else submits job_2, which is identical to job _1. Oracle Reports Services uses 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, then 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.

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/rwcgi60.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 the ORACLE_HOME\REPORT60 directory on the Web server machine.

Usage Notes

The following usage notes might be helpful for key mapping:

- 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, then its defined value is substituted into the command line for Oracle Reports Services. If it is not found, then an error is generated.

- When submitting a URL through 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 the `cgicmd.dat` (Web CGI) file, located in the `ORACLE_HOME\REPORT60` directory, in a text editor.

> **Tip:** Type: `http://your_webserver/cgi-bin/rwcgi60.exe/showmap?` in your Web browser to verify the name of the mapping file that is being used.

2. Add a key mapping entry. A basic key mapping entry looks similar to the following, where `key1` is the name of the key:

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

Except for the special parameters that are described in the file itself, the command line arguments follow the syntax rules of RWCLI60. See Appendix A, "RWCLI60 Command Line Arguments" for more information about the RWCLI60 command line arguments.

If you set the REPORTS60_REPORTS_SERVER environment variable and are sending the request to the default server, then 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, "Specifying a Report Request from a Web Browser".

4.4 Specifying Report Requests

You can specify reports by:

- Building a report
- Specifying a report request from a Web browser
- Scheduling reports to run automatically

4.4.1 Building a Report

To build a report, you do the following:

1. On the machine where your Oracle Reports Services 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 Oracle Reports Services configuration file. See Appendix B, "Oracle Reports Services Configuration Parameters".

The reports source path can also be set in the REPORTS60_PATH environment variable. See Appendix C, "Environment Variables" for more information.

Start the Oracle 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 Oracle Reports Services machine (for example, `C:\WEB_REPORTS`). Refer to the *Building Reports* manual or Oracle Report Builder online help for more information about building a report. To access Oracle Report Builder only help, click on and do the following steps:

---

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" for more information.

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.

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
```

  If you require additional command line arguments, then refer to Appendix A, "RWCLI60 Command Line Arguments" for a list of valid RWCLI60 command line arguments.

- Simplified URL request using key mapping that looks similar to the following:

```
http://your_webserver/cgi-bin/rwcgi60.exe?report=key1
```

If you set the REPORTS60_REPORTS_SERVER environment variable and are sending the request to the default server, then 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. Request as a hyperlink to your Web page your syntax would look 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, then refer to 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 Oracle Reports Queue Manager online help for more information about scheduling your reports.

If you publish your reports on a WebDB site as WebDB component, then you can schedule these report requests to run automatically and push the resulting reports to specified folders on the site. Refer to Chapter 5, "Controlling User Access to Reports" for more information.

5

Controlling User Access to Reports

Access control enables you to restrict user access to reports that are run on Oracle Reports Services. Oracle Reports Services uses WebDB to perform a security check that ensures that users have the necessary privileges to run reports on restricted Oracle Reports Services and printers. Access control determines the following:

- What report definition files, Oracle Reports Services, and printers are restricted.

- Who has access privileges to run requested reports on a restricted Oracle Reports Services and output to a restricted printer.

- When report definition files, Oracle Reports Services, 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 Oracle Reports Services 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 Oracle Reports Services that accepts the request, and any printers that are used to print report output. These access controls are added using Oracle Reports Services Security wizards in WebDB. Only users who have Oracle Reports Services 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. Making 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. Scheduling 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 Oracle Reports Services Security in WebDB. See Section 5.3.7, "Step 7. Setting Parameter Values on the Oracle Reports Services Parameter Form" for more information.

5.1 Access Control Configuration and Setup Overview

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

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

5.1.1 Installing and Configuring Oracle Reports Services Security

To install and configure Oracle Reports Services security you need to configure WebDB for Oracle Reports Services security.

5.1.2 Setting up Access Control

To set up access control, you do the following:

1. Configure Oracle Reports Services for access control.

2. Create user accounts.

3. Optionally, create availability calendars in WebDB.

4. Add access to the printer in WebDB.

5. Add access to Oracle Reports Services in WebDB.

6. Add access to the report definition file in WebDB and create a packaged procedure.

   You can batch register multiple reports in WebDB using the Oracle Reports Services Batch Registering utility. Refer to the Oracle Reports Services Batch Registering Reports technical white paper located on the OTN (`http://technet.oracle.com`).

7. Set parameter values on the Parameter Form.

8. Publish the report request on a WebDB site.

9. Optionally, schedule the report to run and push the output to a WebDB site.

5.2 Installing and Configuring Oracle Reports Security in WebDB

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

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

5.2.1 Step 1. Configuring Oracle Reports Security in WebDB

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

- Install WebDB release 2.2 or later.

- Install Oracle Reports Security feature in WebDB.

- Set the authentication cookie domain.

5.2.1.1 Installing WebDB

Install WebDB into an Oracle 7.3.4, Oracle 8.0.5, or Oracle 8*i* 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 Installing Oracle Reports Services Security Packages in WebDB

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

1. If you want to install just Oracle Reports Services security packages, then 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 installs 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:

     ```
     sqlplus /nolog @rwwwvins.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

Field	Description
username	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.
password	A password that logs you on to the WebDB schema.
database	The name of the database that connects you to the WebDB schema.

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 Setting the Authentication Cookie Domain

You set the authentication cookie domain so that the cookie can send the authentication information to Oracle Reports Services where the report is sent. Click on   and do the following:

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, where `my_company.com` is the domain name of the Oracle Reports Services

   `ORCookieDomain=my_company.com`

2. Save your changes and close the configuration file.

5.3 Setting Up Access Controls in WebDB Example

This example provides step-by-step instructions that will help you configure your Oracle Reports Services for access control. You will add access to the report definition file, Oracle Reports Services, 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:

- Oracle Reports Services is configured using the Web CGI configuration.

- A printer that Oracle Reports Services is installed and running.

- You have WebDB site administrator privileges. This enables the Oracle Reports Services 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 in the `ORACLE_HOME\TOOLS\DOC60\US\RBBR60` directory.

- You must be able to access the Oracle Reports Services demo tables to run the `security.rdf` file on Oracle Reports Services. 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→ProgramsStart→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, then Oracle Reports Services 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. Configuring Oracle Reports Services for Access Control

Oracle Reports Services must be installed and configured before you can perform this step. See Chapter 3, "Configuring the Oracle Reports Server on Windows NT and UNIX" for information.

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

- Create a TNS names alias that connects to WebDB.
- Restrict access to Oracle Reports Services.

5.3.1.1 Creating the TNSnames Alias that Connects to WebDB

You need to create a TNSnames alias for WebDB in the `tnsnames.ora` file on the machine where Oracle Reports Services is installed. This enables Oracle Reports Services to communicate with WebDB. If you installed Oracle Reports Services Security feature from your Oracle Reports Services machine, then a TNSnames alias for WebDB has already been created for you. You can skip this step and go to Section 5.3.1.2, "Restricting Access to Oracle Reports Services".

You can create the TNSnames alias using the Net8 Easy Config tool or by editing the `tnsnames.ora` file in a text editor.

To create the TNSnames alias, you will need the following information:

- 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.

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 Oracle Reports Services 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. If you choose this option, then you can skip steps 2 and 3.

   - Open the `tnsname.ora` file located in the `ORACLE_HOME\NET80\ADMIN` directory. Go to step 2.

2. Add the following TNSnames alias to connect to WebDB:

```
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.
`.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 Restricting Access to Oracle Reports Services

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

To run a report request, you add access to the report definition file in WebDB. If you want to run unrestricted report requests, then ensure the Run Only Registered Report Definition Files option is unchecked in the Server Access wizard in WebDB for this Oracle Reports Services. Users, however, will still need to authenticate themselves to the Oracle Reports Services to run the report.

On Oracle Reports Services machine, do the following:

1. Open the `repserver.ora` configuration file (located in the `ORACLE_HOME\REPORT60\SERVER` directory) in a text editor. Set the SECURTYTNSNAME parameter using the following syntax, where `sec_rep` is the TNSnames alias of the WebDB server instance defined in the `tnsnames.ora` file:

   ```
   SECURITYTNSNAME="sec_rep"
   ```

2. Save and close Oracle Reports Services configuration file.

3. Stop and restart Oracle Reports Services to accept the changes to the Oracle Reports Services configuration file.

5.3.2 Step 2. Creating User Accounts

You will need to create the following user accounts:

- The Oracle Reports Services system administrator to create and maintain access controls for the restricted Oracle Reports Services, report definition files, and printers. See Section 5.3.2.1, "Creating the Oracle Reports Services System Administrator User Account". Once you create the Oracle Reports Services system administrator, you can start creating access controls for Oracle Reports Services, 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 Oracle Reports Services and printer. See Section 5.3.2.2, "Creating Users Accounts for Running Reports". You can create user accounts at anytime. However, if you are restricting access to Oracle Reports Services, report definition files, or printers, and know which users should have access to them, then it is best to create the user's accounts first.

5.3.2.1 Creating the Oracle Reports Services 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 Oracle Reports Services 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 might 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. Click on ❓ and do the following:

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: Packaged procedures and the parameter list of values that you create can be owned by different schemas. You might 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, then 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.

If someone else is the site administrator, then 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 Creating 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. Oracle Reports Services 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, Oracle Reports Services basic user functions (that is, the RW_BASIC_USER role) are implied if users are not assigned specific Oracle Reports Services 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. Click on and do the following:

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.

For this example, 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 Oracle Reports Services roles as needed.

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

5.3.3 Step 3. Creating Availability Calendars

An availability calendar determines when report definition files, Oracle Reports Services, and printers are available for processing. Availability calendars are not necessary if the reports definition files, Oracle Reports Services, and printers are always 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, Oracle Reports Services, or printer. If your production environment requires more than one availability rule, then you will need to combine availability calendars.

In this example, 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 Creating 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.

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

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

2. On the Oracle WebDB home page, click **Administer**. You also can click ![icon] from the navigation toolbar on any WebDB page to access the **Administer** page.

3. On the **Administer** page, click **Reports Developer Security**.

4. On the Oracle Reports Services Security page, click **Availability Calendars**.

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. If the Daily calendar already exists, then append your initials to it (for example, DailyAA).

8. Click ![arrow icon] .

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

4. Click .

5. Follow steps 9 through15 in Section 5.3.3.1, "Creating the Daily Calendar" to define the following availability rule:

Table 5–2   Maintenance Calendar Rule

Field	Value
Duration Start	Specify a date starting on a Saturday (for example, January 8, 2000), and time starting at 3:00 p.m.
Duration End	Specify the same date defined as the start date, and time ending at 10:00 p.m.
Repeat	Choose **Weekly**.

5.3.3.3 Creating 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. If the Christmas calendar already exists, then append your initials to it (for example, ChristmasAA).

4. Click ▶.

5. Follow steps 9 though15 in Section 5.3.3.1, "Creating the Daily Calendar" to define the following rule:

Table 5–3   Christmas Calendar Rule

Field	Value
Duration Start	Specify December 25 and 12:00 a.m.
Duration End	Specify December 26 and 12:00 a.m.
Repeat	Choose **Yearly**.

5.3.3.4 Creating a Combined Availability Calendar

In this example, you 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. 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.

    It is 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. Adding Access to a Oracle Reports Services 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 Oracle Reports Services 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.

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.

2. On the **Reports Developer Security** page, click **Printer Access**.

3. Click **Create** to add printer access to WebDB.

4. On the **Printer Name** page, type Reports_Printer in the **Printer Name** field. 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 Oracle Reports Services 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 🔲 to find the availability calendar. If you want to make this printer available all the time, then do not specify a calendar.

11. Click ▶.

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

5.3.5 Step 5. Adding Access to Oracle Reports Services in WebDB

Oracle Reports Services access defines the following in WebDB:

- What Oracle Reports Services is available in WebDB for processing requests.

- What printer is available to Oracle Reports Services.

- Who has access privileges to send report requests to this Oracle Reports Services.

- When this Oracle Reports Services is available to accept report requests.

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.

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. 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. The Oracle Reports Services TNS name is Oracle Reports Services entry name that is added to the tnsname.ora file when you installed and configured Oracle Reports Services. See Chapter 3, "Configuring the Oracle Reports Server on Windows NT and UNIX" for more information.

6. Type the Oracle Reports Services Web Gateway URL in lowercase:

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

   The Oracle Reports Services Web Gateway URL is determined by the virtual location of the Web CGI.

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

8. Click .

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

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

10. Click ![icon] 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 ![icon] to find the availability calendar. If you want to make this server available all the time, then 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. Adding Access to the Report Definition File in WebDB

Report Definition File access defines the following in WebDB:

- What Oracle Reports Services .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 Oracle Reports Services and printers that are available to users.

In this example, you restrict access to the security.rdf file (located in the ORACLE_HOME\TOOLS\DOC60\US\RBBR60 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.

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

5.3.6.1 Creating 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, then 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 example, you create a list of values for the P_Lastname user parameter that lists the last names of these employees.

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 Oracle Report Builder using the Parameter Form editor. Click on ![icon] and do the following:

---

1. For Oracle 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 ![icon] 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. 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

Display Value	Return Value	Display Order
Abers	Abers	1
Costner	Costner	2
Matsumoko	Matsumoko	3
Williams	Williams	4

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, then you can do so here.

5.3.6.2 Adding Access the Report Definition File

If you back out of the wizard page (that is, click **Back** on your Web browser), then you will lose the settings you defined on that page. If you need to make changes, then 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**.

2. On the **Reports Developer Security** page, click **Report Definition File Access**.

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. The report name cannot be prefaced with numeric characters (for example, 401K_report is an invalid file name and my_401K_report is valid). 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 Oracle Reports Services File Name. Ensure Oracle Reports Services can find this report definition file. The report's source path must be set in the SOURCEDIR parameter in Oracle Reports Services configuration, or the path must be set in the REPORTS60_PATH environment variable.

8. Click arrow.

9. On the **Users and Roles** page, choose JABERS and your Oracle Reports Services system administrator user account from the **All Users** list box to specify who can run this report. Ensure that you choose the same users who have been given access to the printer and Oracle Reports Services.

10. Click ![icon] to move this user to the **Selected Users** list box.

11. Click arrow.

12. On the **Availability Calendar** page, type `Production` as the availability calendar, or click ![icon] to find the availability calendar. If you want to make this report definition file available all the time, then do not specify a calendar.

13. Click arrow.

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. If the printer you defined does not appear, then you might 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. Optionally, choose another **Parameter** form template. The template you choose determines the page style on which the Runtime Parameter Form is displayed.

18. Click arrow.

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. The P_LASTNAME is the name of the parameter defined in report. Open the security.rdf file in Oracle 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 [icon] 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 enables users to specify the database that they can connect to if they want to schedule the report to run automatically.

26. Click [icon] 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 example explains how to set the default parameter values in the Parameter Form that are used to run the report.

    If an invalid package is created, then you will be unable to proceed to the next step. Verify the access controls that you defined for the printer, Oracle Reports Services, 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 [icon] 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. Setting Parameter Values on the Oracle Reports Services Parameter Form

As the Oracle Reports Services 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 Running the Report Output to Cache

In this example 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, "Setting 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.

To access the Manage Component page, click [icon]. 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 Oracle Reports Services Parameter Form, set the following parameters:

Table 5–5   Parameter Form Settings for Debugging Cache Output

Parameter	Value
SERVER	repserver
PRINTER	Reports_Printer
DESTYPE	Cache
DESFORMAT	HTMLCSS
DESNAME	blank
COPIES	1
P_LASTNAME	Abers
P_SSN	559014203
USERID	username/password@my_db
	where username/password@my_db is the user name 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 Running the Report Output to a Restricted Printer (Optional)

Following are the steps you would follow if you want to run report output to a
restricted printer:

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

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

Table 5–6   Parameter Form Settings for Debugging Printer Output

Parameter	Value
SERVER	repserver
PRINTER	Reports_Printer
DESTYPE	Printer
DESFORMAT	PDF
DESNAME	defaults to the printer name
COPIES	1
P_LASTNAME	Abers
P_SSN	559014203
USERID	username/password@my_db
	where `username/password@my_db` is the user name and password for the database you want to connect to.

3. Click **Run Report**.

4. Click **OK** when a message appears stating that the report was printed
   successfully.

5.3.7.3 Setting 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 Oracle Reports Services Parameter Form, set the following parameters:

Table 5–7   Default Parameter Settings for Users

Parameter	Default Value	Visible to User
SERVER	`repserver`	No
PRINTER	Reports_Printer	No
DESTYPE	Cache	Yes
DESFORMAT	HTMLCSS	Yes
DESNAME	blank	No
COPIES	1	Yes
P_LASTNAME	blank	Yes
P_SSN	`Type your SSN`	Yes
USERID	`Type the database logon`	Yes

You might 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. Making 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. Making 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 Creating a WebDB Site

Create a WebDB site if it has not already been created. Click on [icon] and do the following:

---

1. For WebDB online help on this task, click the help button **on the title bar...**

2. **Click [icon] and type...** web site.

3. **Click Find and click...** Creating web sites to display the topic.

To create a WebDB site, the Oracle Reports Services 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, then ask that person to create the WebDB site.

5.3.8.2 Creating 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 Oracle Reports Services 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).

If you make a folder public, then PUBLIC users (that is, users who have not logged on to the WebDB site) can access the report's Parameter/Scheduling form and can 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 saved on the Parameters/Scheduling form is secured and can only be viewed.

Click to access context-sensitive help for the current wizard page. Click on the title bar to access theWebDB 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
    ```

    If you have site administrator privileges to create a site, then click 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 Oracle Reports Services system administrator.

    To add a WebDB component to a WebDB site, your Oracle Reports Services 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, then 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. 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 Adding the Report Request to the Folder

To add the report request to the folder, do the following:

1. On the Benefits page, click ![Add Item] 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 Running the Report as a User

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

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

1. Click **Log Off** on the navigation bar to log off as the Oracle Reports Services 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 5–8   User's Default Parameter Settings

Parameter	Default value
DESTYPE	Cache
DESFORMAT	HTMLCSS
COPIES	1
P_LASTNAME	Abers
P_SSN	559014203
USERID	username/password@my_db
	where `username/password@my_db` is the user name and password for the database you want to connect to.

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

   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, then the default settings defined by the Oracle Reports Services system administrator would display. That user could then personalize her or his own settings.

8. Click **Run Report**.

5.3.9 Step 9. Scheduling 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 reviewed the example in
> Section 5.3.8, "Step 8. Making the Report Available to Users".

5.3.9.1 Creating 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 example) can create her or his own personal folder.

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 [Finish].

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 to push it to the JABERS personal folder.

5.3.9.2 Scheduling the Report

In this example, 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.

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. 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 Section 5.3.8.4, "Running the 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. If you want to run the report immediately, then choose the Immediately option.

6. Choose the following **Repeat** option:

Last Friday of each month on or before the 30 th.

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:

Table 5–9   Output Destination Settings

Field	Value
Site	The name of the WebDB site in which the Result folder is located.
Log File Folder	JABERS
Result Title	My 401K Report
Result Folder	JABERS
Overwrite Previous Result	uncheck
Expiration	60 days

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, then you must type the exact folder name. If you mistype the folder names, then WebDB will not be able to find them, and by default will add the named folders to the Oracle Reports Services Output and Oracle Reports Services 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. If you scheduled the report to run immediately, then 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 Viewing the Pushed Report Output

To view the pushed report output, do the following:

1. Click **Site Map** from the navigation bar. 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 might have running this scheduled report. Depending on the Oracle Reports Services role (for example, RW_BASIC_USER) this user is assigned, you might see different status details. If users are having problems scheduling and running their reports, then they should contact the Oracle Reports Services system administrator for help.

5.3.9.4 Optional Examples

Suppose that the Human Resource director asked you (the Oracle Reports Services 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 examples in this chapter to help you add access to the `template.rdf` report (located in the `ORACLE_HOME\TOOLS\DOC60\US\RBBR60` directory) in WebDB.

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

The **Stock** folder must be set up to display for public users.

5.4 Summary

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

- Create availability calendars to determine when reports, Oracle Reports Services, and printers are available for processing.

- Add access to printers, Oracle Reports Services, 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.

6

Configuring Oracle Reports Services Clusters

This chapter will show you how to configure Oracle Reports Services 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.

Oracle Reports Services 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 Oracle Reports Services to grow as your organization grows.

Before you begin to configure your Oracle Reports Services for clustering, you should be familiar with the basic Oracle Reports Services architecture. See Chapter 1, "Publishing Architecture and Concepts" for more information. You must also have already set up your Oracle Reports Services using a basic configuration. See Chapter 3, "Configuring the Oracle Reports Server on Windows NT and UNIX" for more information.

6.1 Clustering Overview

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

Table 6–1   Example Server Machines Descriptions

Machine/Server TNS name	Description	Master/Slave
NT-1	4 CPU NT server	Master
NT-2	2 CPU NT server	Slave
SUN-1	2 CPU Sun Solaris workstation	Slave

> **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 Oracle Reports Services in a cluster as described in this overview, see Section 6.2, "Configuring Oracle Reports Services 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 parameter and MINENGINE parameter for the master and each slave server are set to the number of CPUs available on each machine.

Once the machines are 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).

It is possible for slave servers to remain fully functional Oracle Reports Services 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 Oracle Reports Services configuration are set to three. This means that three engines are dedicated to the NT-2 Oracle Reports Services 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 Oracle Reports Services 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 Oracle Reports Services in a Cluster Example

This section provides step-by step instructions for configuring Oracle Reports Services 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 Services component has been installed.

- Oracle Reports Services has been configured using the machine name as the TNS service entry name (for example, NT-1) in the `tnsames.ora` file.

- 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) you open the `tnsnames.ora` located in the `ORACLE_HOME\NET80` directory in a text editor, and add the following. The `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:

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

On the NT-2 machine (slave) you do the following:

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

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

2. Open the `nt-2.ora` (the Oracle Reports Services 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 your 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:

    - Engine settings are defined that identify 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.

The ENGLIFE and MAXIDLE parameters for the master server's engines are implied for all slave engines. The unit of measure for the MAXIDLE parameter is minutes and the ENGLIFE parameter is the number of engines.

On the NT-1 server machine (master) you do the following:

1. Open `nt-1.ora` (the Oracle Reports Services 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
    ```

    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.

Usage Notes

When configuring the master server, you should consider the following:

- Each slave definition must be surrounded by parenthesis.

- 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).

- 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 Oracle Reports Services 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.

If you set the REPORTS60_REPORTS_SERVER environment variable to the master server, then 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, then the Retry settings defined in the SCHEDULE command line argument dictate whether the job will be re-run. 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 might not solve the problem. The resubmitted job might 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 6–2   Additional Server Machine Description

Machine/Server TNS name	Description	Master/Slave
SUN-2	4 CPU Sun Solaris server	Slave

This example assumes that this machine has already been configured as an Oracle Reports Services. The TNS service entry name for Oracle Reports Services is the machine name.

On the SUN-2 server machine (slave), open the `sun-2.ora` (the Oracle Reports Services configuration file) located in the `ORACLE_HOME\REPORT60\SERVER` directory and add the following, 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:

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

On the NT-1 server machine (master), do the following:

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

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

2. Open the `nt-1.ora` (the Oracle Reports Services configuration file) and add the following bold text to the already existing CLUSTERCONFIG parameter:

```
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 Oracle Reports Services 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 because it was down.

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

7

Customizing Reports at Runtime

Oracle Reports Services can run report definitions built with XML tags and merge them with other report definitions. In previous releases, a report had to be built and saved in the Oracle Report Builder in order to be run by Oracle Reports Services. With the 6*i* release, 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 to a file. As a result, you can use XML report definitions to make batch updates to 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 Oracle 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 Oracle Report Builder. If you can generate XML tags, then you can create a report definition that can be run by Oracle Reports Services.

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.
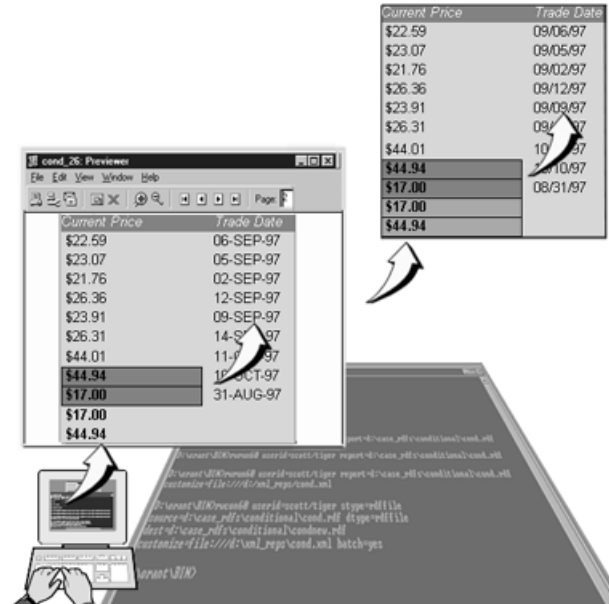
In order to be run by itself, an XML report must contain a full report definition. As shown in the figure below, a self-contained XML report is one that is run without being applied to another report.
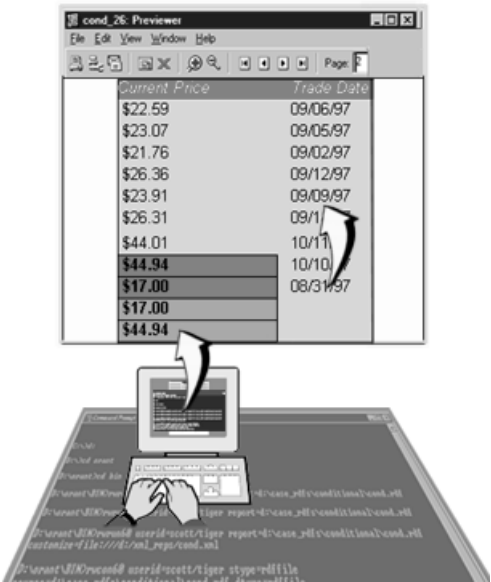
7.1.1 Creating and Using XML Report Definitions

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.[1] 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>
```

This sample would change the formatting characteristics of some fields when applied to another report. 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 Oracle Reports Services.[2]

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 Oracle Reports Runtime and Oracle 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:

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

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

```
<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>
-

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 Oracle Report Builder. The finished .RDF files for these reports are located in the 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 Services 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 7–1   XML Report Definitions for Building Reports

.XML File	.RDF File	Description
cond.xml	cond.rdf	cond.xml changes: ■ The format mask of F_trade_date to MM/DD/RR. ■ The fill colors of F_Mincurrent_pricePersymbol and F_Maxcurrent_pricePersymbol. cond.xml adds: ■ HTML in the report escapes to be inserted when generating HTML output. For more information, refer to Section 7.2.2.1, "Formatting Modifications Example".
temp.xml	temp.rdf	temp.xml changes: ■ The field labels for F_high_365 and F_low_365 temp.xml adds: ■ A formatting exception to F_p_e to highlight values greater than 10 ■ A formatting exception to F_p_e1 to highlight values greater than 10 For more information, refer to Section 7.2.2.2, "Formatting Exception Example".
sect.xml	sect.rdf	sect.xml adds: ■ Program units to the report ■ Link destinations to the detail records in the main section of the report ■ Hyperlinks from the employee summary in the header section to the detail records in the main section For more information, refer to Section 7.2.2.3, "Program Unit and Hyperlink Example".

Table 7–1   (Cont.) XML Report Definitions for Building Reports

.XML File	.RDF File	Description
ref.xml	ref.rdf	ref.xml adds: ■ A new query, Q_summary, to the data model ■ A header section to the report that uses the data from the new query, Q_summary For more information, refer to Section 7.2.2.4, "Data Model and Formatting Modifications Example".

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 the cond.xml file modifies some basic formatting characteristics of the cond.rdf file and adds some HTML code to be inserted at the beginning and end of the report when generating HTMLCSS output.

Tips on this Example

■ 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.

- The `header_example.html` file 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).

```
<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\rbbr60\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>
                      </center>
                      </body>
                      </html>
                   ]]>
                </property>
               </properties>
             </object>
           </customize>
         </report>
```

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. If you are creating a new field, then you can specify the label using the label attribute of the <field> tag.

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

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.

- 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 see 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>
   <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>
</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

- 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.

- Notice 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, then 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, then you could place it inside of the 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 Oracle 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

- 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. 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 Oracle 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 linked 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.

```xml
<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,
           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>
            <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>
                <group>
                  <field name="f_city1" source="city1" label="City"
                        font="Arial" fontSize="8"
                        formatTrigger="F_city1FormatTrigger">
                    <labelAttribute font="Arial" fontSize="8"
                                    fontStyle="bold" textColor="yellow"/>
                  </field>
                  <field name="f_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="LNNNGNNNGNNNGNN0D00">
                    <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="f_Profit_Margin" source="Profit_Margin"
                        label="Margin%" font="Arial" fontSize="8"
                        formatMask="N0%">
                   <labelAttribute font="Arial" fontSize="8"
                                   fontStyle="bold" textColor="yellow"/>
                </field>
             </group>
          </groupLeft>
         </section>
         <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="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="LNNNGNNNGNNNGNN0D00">
                <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="LNNNGNNNGNNNGNN0D00">
                <labelAttribute font="Arial" fontSize="8"
                                fontStyle="bold" textColor="black"/>
             </field>
           </group>
```

```
                <group>
                   <field name="f_state" source="state" label="State:"
                          font="Arial" fontSize="8">
                      <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="LNNNGNNNGNNNGNN0D00">
                      <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="LNNNGNNNGNNNGNN0D00">
                      <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="LNNNGNNNGNNNGNN0D00">
                      <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="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>
  <function name="F_city1FormatTrigger">
```

```
        <![CDATA[
           function F_city1FormatTrigger return boolean is
             begin
               SRW.SET_HYPERLINK('#QTR_CITY_DETAILS_&<' || LTRIM(:quarter1)
||
                                   LTRIM(:city1) || '>');
               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 RWCLI60, 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 sends a job request to Oracle Reports Services 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 destype=file desname=emp.pdf desformat=PDF
  server=repserver
```

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

```
rwrun60 userid=username/password@mydb report=emp.rdf
  customize=e:\myreports\emp.xml destype=file desname=emp.pdf
  desformat=PDF
```

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 sends a job request to Oracle Reports Services 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\emp1.xml)"
  userid=username/password@mydb destype=file desname=emp.pdf desformat=PDF
  server=repserver
```

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

```
rwrun60 report=emp.rdf
  customize="(e:\corp\myreports\emp0.xml,
  e:\corp\myreports\emp1.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.

The `videosales_custom.rdf` file 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

- 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. The `hilite_profits`, `hilite_costs`, and `hilite_sales` parameters 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 might need to spread very large XML report definitions across several columns. If so, then you might have to create several definitions in memory and apply them separately rather than creating one large definition and applying it once.

```
function AfterPForm return boolean is
begin
SRW.ADD_DEFINITION('<report name="vidsales_masks"
author="Generated" DTDVersion="1.0">');
IF :MONEY_FORMAT='$NNNN.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"/>');
 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="LNNNNNNNNNNNN0D00"/>');
          SRW.ADD_DEFINITION('    <field name="F_SumTOTAL_COSTPerCITY"
                     source="SumTOTAL_COSTPerCITY"
          formatMask="LNNNNNNNNNNNN0D00"/>');
          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" formatMask="LNNNNNNNNNNNN0"/>');
          SRW.ADD_DEFINITION('    <field name="F_TOTAL_SALES"
                     source="TOTAL_SALES" formatMask="LNNNNNNNNNNNN0"/>');
          SRW.ADD_DEFINITION('    <field name="F_TOTAL_COST"
                     source="TOTAL_COST" formatMask="LNNNNNNNNNNNN0"/>');
          SRW.ADD_DEFINITION('    <field name="F_SumTOTAL_PROFITPerCITY"
                     source="SumTOTAL_PROFITPerCITY"
          formatMask="LNNNNNNNNNNNN0"/>');
          SRW.ADD_DEFINITION('    <field name="F_SumTOTAL_SALESPerCITY"
                     source="SumTOTAL_SALESPerCITY"
          formatMask="LNNNNNNNNNNNN0"/>');
          SRW.ADD_DEFINITION('    <field name="F_SumTOTAL_COSTPerCITY"
                     source="SumTOTAL_COSTPerCITY" formatMask="LNNNNNNNNNNNN0"/>');
          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(' <layout>');
            SRW.ADD_DEFINITION('  <section name="main">');
            SRW.ADD_DEFINITION('     <field name="F_TOTAL_COST"
                     source="TOTAL_COST">');
            SRW.ADD_DEFINITION('      <exception textColor="red">');
            SRW.ADD_DEFINITION('       <condition source="TOTAL_COST"
                     operator="gt" operand1=":hilite_costs"/>');
            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('   </field>');
  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 Oracle Reports Services to run a report, emp.xml, by itself:

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

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

```
rwrun60 userid=username/password@mydb
  report=e:\corp\myreports\emp.xml
  destype=file desname=emp.pdf desformat=PDF
```

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, then 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 Oracle 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
.
.
.
+-----------------------------------+
| Report Builder Profiler statistics  |
+-----------------------------------+
          TOTAL ELAPSED Time:      11.00 seconds
        Reports Time:      10.00 seconds (90.90% of TOTAL)
              ORACLE Time:       1.00 seconds ( 9.09% of TOTAL)
                      UPI:       0.00 second
                      SQL:       1.00 seconds
 TOTAL CPU Time used by process: N/A
```

7.4.3 RWBLD60

When designing an XML report definition, it is sometimes useful to open it in Oracle Report Builder. In Oracle 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, then opening the definition in Oracle 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 Oracle 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 Oracle 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 Oracle 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_DEFINTION to build an XML report definition in memory, then 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.

```
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;
```

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:

```
custom_summary  Text_IO.File_Type;
```

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:

```
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 Services 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. 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

<!-- --> tag enables you to include comments within your XML report definition. The parser will ignore any text between the comment delimiters. If you are using PL/SQL (SRW.ADD_DEFINITION) to build your XML report definition, then you can incorporate comments in the program unit using the PL/SQL comment delimiters (for example, -- or /* */).

Syntax

```
<!--
    comment_content
-->
```

Example

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

```
<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 might 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.

```
<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>
```

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.

Refer to Section 7.5.7, "<exception>" for more information.

Syntax
```
<condition
  source="source_column_name"
  operator="eq | lt | lteq | neq | gt | gteq | btw | notBtw | like | notLike
     | null | notNull"
  [operand1="comparison_value"]
  [operand2="comparison_value"]
  [relation="and | or"]
/>
```

Attributes

The following table describes the attributes of the <condition> tag:

Table 7–2   <condition> Tag Attributes

Attribute	Required or Optional	Description
source	Required	Is the name of the source column to be used in the condition.
operator	Required	Is the operator to use in comparing other values to the source column. ■ eq (equal) ■ lt (less than) ■ lteq (less than or equal) ■ neq (not equal) ■ gt (greater than) ■ gteq (greater than or equal) ■ btw (between) ■ notBtw (not between) ■ like ■ notLike ■ null ■ notNull
operand1	Optional	Is the value to which the source column is being compared. If the operator is null or notNull, then no operands are required. If the operator is btw or notBtw, then you must also specify operand2.
operand2	Optional	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)

Table 7–2   (Cont.) <condition> Tag Attributes

Attribute	Required or Optional	Description
relation	Optional	Defines whether there are multiple conditions and, if there are, how they should be related.
		▪ The and means that the formatting exception is applied only if both are met.
		▪ The or means that the formatting exception is applied if either condition is met.

Usage Note

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.

```
<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>
```

Examples

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

```
<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>
      </center>
      </body>
      </html>
     ]]>
    </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)
    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 Services (SQL and Express) are supported by this tag.

Syntax
```
<dataSource>
   content_of_data_source
</dataSource>
```

Examples

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

```
<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 7–3   <exception> Tag Attributes

Attribute	Required or Optional	Description
lineColor	Optional	Is the name of the border color to apply when the condition is met. If noLine is specified, then the border is transparent (that is, invisible).
fillColor	Optional	Is the name of the fill color to apply when the condition is met. If noFill is specified, then the background is transparent.
textColor	Optional	Is the name of the text color to apply when the condition is met.

Table 7–3   (Cont.) <exception> Tag Attributes

Attribute	Required or Optional	Description
hide	Optional	Is whether to hide the field when the condition is met. ■ A yes means the field is hidden when the condition is met. ■ A no means the field is not be hidden when the condition is met.
font	Optional	Is the name of the font to apply when the condition is met.
fontSize	Optional	Is the size of the font to be used when the condition is met.
fontStyle	Optional	Is the style of the font to be used when the condition is met: ■ regular ■ italic ■ bold ■ boldItalic
fontEffect	Optional	Is the effect of the font to be used when the condition is met: ■ regular ■ strikeout ■ underline ■ strikeoutUnderline

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, then 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.

```
<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.

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.

```
<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

```
<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"]
  [fontEfffect="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 7–4   <field> Tag Attributes

Attribute	Required or Optional	Description
name	Required	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, then the attributes specified will override those in the .RDF file.
source	Required, for creating new fields Optional, for modifying existing fields	Is the source column from which the field gets its data. The source column must exist in the data model.

Table 7–4   (Cont.) <field> Tag Attributes

Attribute	Required or Optional	Description
label	Optional	Is the boilerplate text to be associated with the field. To control the formatting attributes of the label, you must use the <labelAttribute> tag. For more information refer to Section 7.5.15, "<labelAttribute>".
		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 <object> tag. For more information, refer to Section 7.5.22, "<object>".
currency	Optional	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.
tsep	Optional	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.
formatTrigger	Optional	Is the name of a PL/SQL program unit that is to be used as the format trigger for the field. Format triggers must be functions. For more information refer to the Oracle Report Builder online help system and look for format trigger in the index.
font	Optional	Is the name of the font to be used for the field contents.
fontSize	Optional	Is the size of the font to be used for the field contents.
fontStyle	Optional	Is the style of the font to be used for the field contents: ■ regular ■ italic ■ bold ■ boldItalic

Table 7–4 (Cont.) <field> Tag Attributes

Attribute	Required or Optional	Description
fontEffect	Optional	Is the effect of the font to be used for the field contents: ■ regular ■ strikeout ■ underline ■ strikeoutUnderline
lineColor	Optional	Is the name of the color to be used for the border of the field. If noLine is specified, then the field's border is transparent (that is, invisible).
fillColor	Optional	Is the name of the color to be used as the background for the field. If noFill is specified, then the background is transparent.
textColor	Optional	Is the name of the color to be used for the field contents.
alignment	Optional	Is how the text should be justified within the field: ■ start ■ left ■ center ■ right ■ end
hyperlink	Optional	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.
linkdest	Optional	Is the target to be used when hyperlinking to this field's contents. This attribute is only used when generating HTML or PDF output. It is ignored for other types of output such as PostScript or ASCII.
formatMask	Optional	Is the mask to be applied when displaying the field's contents. For more information on the format mask syntax, refer to the Oracle Report Builder online help system and look under format mask in the index.

Examples

The following example shows a section in the layout of a report definition that
defines fields within two break groups for a matrix report:

```
<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="LNNNGNNNGNNNGNN0D00">
            <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="LNNNGNNNGNNNGNN0D00">
            <labelAttribute font="Arial" fontSize="8"
                            fontStyle="bold" textColor="black"/>
    </field>
</group>
<group>
    <field name="f_state" source="state" label="State:"
            font="Arial" fontSize="8">
        <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. 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, then you must also nest <field> tags to list the fields you want to include in the form layout.

Refer to Section 7.5.8, "<field>" for more information on the <field> tag

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:

```
<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), then you can use the <summary> tag, which requires no PL/SQL.

Refer to Section 7.5.29, "<summary>" for more information.

Syntax

```
<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 7–5   <formula> Tag Attributes

Attribute	Required or Optional	Description
name	Required	Is the name of the formula column.
source	Required	Is the name of a PL/SQL function defined within the <programUnits> tag that performs the desired operation for the formula.
dataType	Optional	Is the type of data that will be generated by the formula. For example, if the formula performs a mathematical operation, then the result is a number. The possible values for dataType are: ■ number ■ character ■ date
width	Optional	Is the number of characters wide of the result of the formula.

Example

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

```
<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>
 <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 7–6   <function> Tag Attributes

Attribute	Required or Optional	Description
name	Required	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 <field> tag).

Example

The following example shows a segment of an XML report definition that defines some PL/SQL functions. The functions are referenced from fields in the layout through the formatTrigger attribute.

```
<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.

```
<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="LNNNGNNNGNNNGNN0D00">
            <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="LNNNGNNNGNNNGNN0D00">
            <labelAttribute font="Arial" fontSize="8"
                            fontStyle="bold" textColor="black"/>
    </field>
</group>
<group>
    <field name="f_state" source="state" label="State:"
            font="Arial" fontSize="8">
          <labelAttribute font="Arial" fontSize="8"
                          fontStyle="bold" textColor="black"/>
    </field>
</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, then 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"/>
     <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"/>
    </groupAbove>
</section>
```

7.5.14 <groupLeft>

Description

The <groupLeft> tag delimits a master-detail style within a section of the report's layout. The master records are placed to the left of the detail records. If you use the <groupLeft> tag, then 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"/>
    <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>
```

7.5.15 <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, then 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"]
  [fontEfffect="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

Attribute	Required or Optional	Description
font	Optional	Is the name of the font to be used for the field label.
fontSize	Optional	Is the size of the font to be used for the field label.

Table 7–7   (Cont.) <labelAttribute> Tag Attributes

Attribute	Required or Optional	Description
fontStyle	Optional	Is the style of the font to be used for the field label: ■ regular ■ italic ■ bold ■ boldItalic
fontEffect	Optional	Is the effect of the font to be used for the field contents: ■ regular ■ strikeout ■ underline ■ strikeoutUnderline
lineColor	Optional	Is the name of the color to be used for the border of the field. If noLine is specified, then the field's border is transparent (that is, invisible).
fillColor	Optional	Is the name of the color to be used as the background for the field. If noFill is specified, then the background is transparent.
textColor	Optional	Is the name of the color to be used for the field contents.
alignment	Optional	Is how the text should be justified within the field: ■ start ■ left ■ center ■ right ■ end

Example

The following example shows a segment of an XML report definition that defines a section with a group left layout inside of it. 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.

```
<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. This is not a complete layout model and would have to be applied as a customization to an .RDF file:

```
<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>
```

The following example shows another layout segment of an XML report definition. 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.

```
<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="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="LNNNGNNNGNNNGNN0D00">
       <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="LNNNGNNNGNNNGNN0D00">
       <labelAttribute font="Arial" fontSize="8"
                       fontStyle="bold" textColor="black"/>
    </field>
  </group>
  <group>
    <field name="f_state" source="state" label="State:"
            font="Arial" fontSize="8">
      <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="LNNNGNNNGNNNGNN0D00">
       <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="LNNNGNNNGNNNGNN0D00">
       <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="LNNNGNNNGNNNGNN0D00">
             <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="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>
```

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

```
<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 7–8   <link> Tag Attributes

Attribute	Required or Optional	Description
parentGroup	Required for group links Optional for column links	Is the name of the parent group that you want to relate to the child query.
parentColumn	Required for column links Ignored for group links	Is the name of a column in the parent query that relates to a column in the child query (that is, child column).
childQuery	Required for group links Optional for column links	Is the name of the child query that relates to the parent group.
childColumn	Required for column links Ignored for group links	Is the name of a column in the child query that relates to a column in the parent query (that is, parent column).

Table 7–8 (Cont.) <link> Tag Attributes

Attribute	Required or Optional	Description
condition	Required	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.)
sqlClause	Required	Is the type of SQL clause that relates the parent group to the child query. The default is a WHERE clause.

Example

The following example shows the data model segment of a report definition with a link between two queries:

```
<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>
```

```
<link        parentColumn="deptno_dept"
             childColumn="deptno_emp"
             condition="eq"
             sqlClause="where"/>
</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, then 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.

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

```
<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:

```
<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="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="LNNNGNNNGNNNGNN0D00">
      <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="LNNNGNNNGNNNGNN0D00">
      <labelAttribute font="Arial" fontSize="8"
                      fontStyle="bold" textColor="black"/>
    </field>
 </group>
```

```
<group>
   <field name="f_state" source="state" label="State:"
          font="Arial" fontSize="8">
     <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="LNNNGNNNGNNNGNN0D00">
     <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="LNNNGNNNGNNNGNN0D00">
     <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="LNNNGNNNGNNNGNN0D00">
     <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="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>
```

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="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>
```

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
</matrixol>
```

Example

The following example shows a segment of an XML report definition that defines the column dimension of a matrix layout:

```
<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="LNNNGNNNGNNNGNN0D00">
      <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="LNNNGNNNGNNNGNN0D00">
      <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="LNNNGNNNGNNNGNN0D00">
   <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>
```

Example

The following example shows a segment of an XML report definition that defines the row dimension of a matrix layout:

```
<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
```
<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 7–9   <object> Tag Properties

Attribute	Required or Optional	Description
name	Required	Is the identifier for the object to which you want to apply the properties.
type	Required	Is the kind of object to which you want to apply the properties: ■ REP_REPORT is the report itself. ■ REP_GROUP is a group in the data model of the report. ■ REP_COL_MAP is a column in the data model of the report. ■ REP_GRAPHIC_TEXT is a boilerplate object in the layout of the report.

Examples

The following example shows a segment of an XML report definition that defines some object properties:

```
<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>
      </center>
      </body>
      </html>
     ]]>
    </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>
```

7.5.23 <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.

Refer to Section 7.5.11, "<function>" for more information.

Syntax

```
<programUnits>
    program_unit_definitions
</programUnits>
```

Example

The following example shows a segment of an XML report definition that defines some PL/SQL. The <programUnits> tag is outside of the <layout> tag and that the functions are referenced from fields in the layout through the formatTrigger attribute.

```
<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>
```

Examples

The following example shows a segment of an XML report definition that defines an object's properties:

```
<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>
      </center>
      </body>
      </html>
     ]]>
    </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>
```

7.5.25 <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

```
<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 7–10   <property> Tag Attributes

Attribute	Required or Optional	Description
name	Required	Is the name of the property that you want to specify. The available properties vary depending upon the type of object. Refer to the **"Usage Notes"** for more information.

Usage Notes

The following table lists the properties that are available for each type of object:

Table 7–11   Valid Properties for Object Types

Object	Valid Properties
Report object (REP_REPORT)	■ xmlTag
	■ xmlAttribute
	■ xmlSuppress
	■ prologType
	■ prolog
	■ beforeReportValue
	■ beforeReportType
	■ afterReportValue
	■ afterReportType
	■ beforePageValue
	■ beforePageType
	■ afterPageValue
	■ afterPageType
	■ beforeFormValue
	■ beforeFormType
	■ afterFormValue
	■ afterFormType
	■ pageNavigationControlValue
	■ pageNavigationControlType
Group object (REP_GROUP)	■ xmlTag
	■ xmlAttribute
	■ outerXMLTag
	■ outerXMLAttribute
	■ xmlSuppress
Column object (REP_COL_MAP)	■ xmlTag
	■ xmlAttribute
	■ XMLSuppress
	■ containXML

Table 7–11   (Cont.) Valid Properties for Object Types

Object	Valid Properties
Boilerplate object (REP_GRAPHIC_TEXT)	■ textSegment

Examples

The following example shows a segment of an XML report definition that defines
an object's properties.

```
<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>
      </center>
      </body>
      </html>
     ]]>
    </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.

```
<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\TOOLS\DOC60\US\RBBR60.
     If it it 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>
    <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>
```

```
    </center>
    </body>
    </html>
   ]]>
  </property>
 </properties>
 </object>
 </customize>
</report>
```

Attributes

The following table describes the attributes of the <report> tag:

Table 7–12   <report> Tag Attributes

Attribute	Required or Optional	Description
name	Optional	Records the name of the report. If the name is not specified, then the default is UNTITLED. If you plan to apply the report definition to an .RDF file, then this name should be the same as the file name without the .RDF extension.
dtdVer	Required	Records the version of the Oracle 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.
title	Optional	Places the specified title at the beginning of the report. When applying the definition title at an .RDF file, this title overrides the existing report title.
author	Optional	Records the name of the author.

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 might have up to three sections in its layout.

For each section, you might 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 7–13   <section> Tag Attributes

Attribute	Required or Optional	Description
name	Required	Is the section's name: header, main, or trailer.
width	Optional	Is the width of one physical page (including the margin) in the unit of measurement of the report (for example, 8.5 inches).
height	Optional	Is the height of one physical page (including the margin) in the unit of measurement of the report (for example, 11 inches).

Example

The following is an example of a <section> definition:

```
<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

```
<select>
   content_of_SELECT
</select>
```

Examples

The following example shows the data source 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>
```

A user parameter is automatically generated for you if you include it as a bind reference in a SELECT statement. For example:

```
<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:

```
<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, then you can use the <formula> tag to create a formula column that uses PL/SQL to perform more complex calculations.

Refer to Section 7.5.10, "<formula>" for more information.

Syntax
```
<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 7–14   <summary> Tag Attributes

Attribute	Required or Optional	Description
source	Required	Is the name of the column whose values are summarized.

Table 7–14   (Cont.) <summary> Tag Attributes

Attribute	Required or Optional	Description
function	Optional	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.
compute	Optional	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.

Table 7–14 (Cont.) <summary> Tag Attributes

Attribute	Required or Optional	Description
reset	Optional	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.
productOrder	Optional	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.
nullval	Optional	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 is displayed for null values fetched for the column. If left blank, then no substitution is done for null values.

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

Optional Attribute	Default Value
function	sum
compute	The parent group of the summary column's group
reset	The parent group of the summary column's group

Table 7–16   Default Values for Summaries in a Matrix Report

Optional Attribute	Default Value
function	sum
compute	The cross product group
productOrder	■ The group containing the summary (for dimension summaries) ■ A list of groups that define the matrix row (for cell summaries)
reset	The highest frequency group of the productOrder

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. 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.

```
<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, then you must also nest <field> tags to list the fields you want to include in the tabular layout.

Refer to Section 7.5.8, "<field>" for more information.

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:

```
<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>
```

Part II

Appendixes

A

RWCLI60 Command Line Arguments

This appendix contains descriptions of RWCLI60 command line arguments. RWCLI60 parses and transfers the command line to the specified Oracle Reports Services (RWMTS60). It uses a command line very similar to RWRUN60.

A.1 Syntax

Following is the syntax for the RWCLI60 command line, where `keyword=value` is a valid command line argument:

```
RWCLI60 MODULE|REPORT=runfile USERID=userid
[ [keyword=]value|(value1, value2, ...) ] SERVER=tnsname
```

A.2 Usage Notes

The following usage notes apply to the RWCLI60 command line:

- 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=, then the command file is read and appended to the original command line before being sent to Oracle Reports Services. 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 Oracle Reports Runtime searches first for a file with extension .REP, then extension .RDF, then .XML, and then no extension. Oracle Reports Runtime will use its file path search order to find the file.

USERID

Description   USERID is your ORACLE user name or placeholder user name (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   The following restrictions apply:

- A command file might 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 might 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   `[TERM=]termtype`

Values   Any valid terminal type.

Default Installation dependent. (See your Oracle Reports Services system administrator for a compatible definition.)

Usage Note The argument or arguments for this keyword might 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 `[ARRAYSIZE=]n`

Values A number from 1 through 9,999. This means that Oracle 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 *Oracle8i 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

CACHE	Sends the output directly to Oracle Reports Services cache. DESTYPE=CACHE is not compatible with the DISTRIBUTE keyword. If the server encounters DISTRIBUTE on the command line, then it is ignored the DESTYPE=CACHE command line argument.
LOCALFILE	Sends the output to a file on the client machine and forces a synchronous call, regardless of the BACKGROUND value.
FILE	Sends the output to the file on the server machine named in DESNAME.
PRINTER	Sends the output to the printer on the server machine named in DESNAME. You must have a printer that Oracle Reports Services can recognize installed and running.

MAIL	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.
SYSOUT	Sends the output to the client machine's default output device and forces a synchronous call.

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 by 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 user names by enclosing the names in parentheses and separating them by commas (for example, (name, name, . . .name)).

Syntax   [DESNAME=]desname

Values   Any valid file name, 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   The following usage notes apply:

- This keyword is ignored if DESTYPE is SCREEN.

- If DESTYPE is PREVIEW, then Oracle Report Builder uses DESNAME to determine which printer's fonts to use to display the output.

- The argument or arguments for this keyword might be case sensitive, depending on your operating system.

In some cases, this parameter might 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, dflt, wide. Ask your System Administrator for a list of valid destination formats.

PDF	Means that the report output is 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.
HTML	Means that the report output is sent to a file that can be read by an HTML 3.0 compliant browser (for example, Netscape 2.2).
HTMLCSS	Means that the report output sent to a file includes style sheet extensions that can be read by an HTML 3.0 compliant browser that supports cascading style sheets.
HTMLCSSIE	Means that the report output sent to a file includes style sheet extensions that can be read by Microsoft Internet Explorer 3.x.
RTF	Means that the report output is sent to a file that can be read by standard word processors (such as Microsoft Word). When you open the file in MS Word, you must choose **View**→**Page Layout** to view all the graphics and objects in your report.
DELIMITED	Means that the report output is sent to a file that can be read by standard spreadsheet utilities, such as Microsoft Excel. If you do not choose a delimiter, then the default delimiter is a TAB.
XML	Means that the report output is 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.

Default   Taken from the Initial Value property of the DESFORMAT parameter. For bit-mapped Oracle 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   The following usage notes apply:

- This keyword is ignored if DESTYPE is SCREEN.

- The value or values for this keyword might 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   The following usage notes apply:

- 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, then it is preferable to set this value to NO. Setting the value to NO, however, might decrease performance, as the LOB might 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   The following usage notes apply:

- This keyword is ignored if DESTYPE is not Printer.

- If COPIES is left blank on the Runtime Parameter Form, then 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 overrides 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   The following usage notes apply:

- A THOUSANDS value entered on the Parameter property sheet overrides 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, then "," 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   The following usage notes apply:

- 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, then "." 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 *Oracle8i 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.

Restriction   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, then output is appended to it. This keyword is only used in character mode.

Syntax   `[LOGFILE=]logfile`

Values   Any valid file name.

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 might be lower.

Default   640K

Usage Note   If this setting is changed in the middle of you session, then the changes does not take effect until the next time the report is run.

BATCH

Description   If BATCH is specified, then 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, then the report is 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 and height depends upon the unit of measurement. For inches, the maximum width and 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: (default page size * character page size)/default character page size. For example, if the character page size is 80 x 20, then the bit-mapped page size would be: (8.5 * 80)/80 x (11 * 20)/66 = 8.5 x 3.33.

Usage Notes   The following usage notes apply:

- On some printers the printable area of the physical page is restricted. For example, the sheet of paper a printer takes might 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 Oracle Report Builder that is bigger than the printable area your printer allows, then clipping might 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 overrides 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 file name, then Oracle 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 Oracle Report Builder Time and ORACLE Time.

- Time is the amount of time spent in Oracle 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 Oracle Report Builder time includes the database time because the database is included in Oracle Report Builder's process.

Syntax   [PROFILE=]profiler_file

Values   Any valid file name 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. Oracle Report Builder makes 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 Services) 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 Services) 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 run at runtime. KEYIN is used to run the keystroke files created with KEYOUT. Since KEYIN is used to run a keystroke file, it is only relevant when running in a character-mode environment.

Syntax `[KEYIN=]keyin_file`

Values Any valid key file name in the current directory.

KEYOUT

Description   KEYOUT is the name of a keystroke file in which you want Oracle Reports Runtime to record all of your keystrokes. You can then use KEYIN to run 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 file name.

ERRFILE

Description   ERRFILE is the name of a file in which you want Oracle Report Builder to store error messages.

Syntax `[ERRFILE=]error_file`

Values Any valid file name.

LONGCHUNK

Description   LONGCHUNK is the size (in kilobytes) of the increments in which Oracle Report Builder retrieves a LONG column value. When retrieving a LONG value, you might 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 might 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   The following usage notes apply:

- 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, then 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 Oracle 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), then you could invoke character-mode RWRUN60 and run the report with MODE=BITMAP. On Windows, specifying MODE=CHARACTER means that the Oracle Report Builder ASCII driver is used to produce editable ASCII output.

Syntax   [MODE=]{BITMAP|CHARACTER|DEFAULT}

Values   The following values apply:

- 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   The following usage notes apply:

- 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 does not appear, regardless of PRINTJOB.
- When DESTYPE=MAIL, the Print Job dialog box does not appear, regardless of PRINTJOB.

TRACEFILE

Description   TRACEFILE is the name of the file in which Oracle Report Builder logs trace information.

Syntax   [TRACEFILE=]tracefile

Values   Any valid file name.

Usage Notes   The following usage notes apply:

- 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 is 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 Oracle 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   The following values apply:

- 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 past 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. 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 Oracle Reports Runtime is fetching data from the database.

Syntax `[NONBLOCKSQL=]{YES|NO}`

Values YES means that other programs can run while data is being fetched. NO means that other programs cannot run 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:

- The Page Navigation Control Type and Page Navigation Control Value properties in the Report Property Palette.

- 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 Oracle Reports Services.

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 Oracle 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 shows the report name as the job name.

Syntax   `[JOBNAME=]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:

```
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
```

> **Note:**   Earlier forms of the SCHEDULE syntax are supported, but only the current SCHEDULE syntax is documented here.

Syntax   Following is the correct syntax:

```
[SCHEDULE=]string
```

where the string is:

```
[FREQ from] TIME [retry {n} + after LEN]
```

FREQ	hourly | daily | weekly | monthly | {every LEN | DAYREPEAT}} | {last {WEEKDAYS | weekday | weekend} before {n}+}
LEN	{n}+ {minute[s] | hour[s] | day[s] | week[s] | month[s]}
DAYREPEAT	{first | second | third | fourth | fifth} WEEKDAYS of month

FREQ	hourly | daily | weekly | monthly | {every LEN | DAYREPEAT}} | {last {WEEKDAYS | weekday | weekend} before {n}+}
WEEKDAYS	mon | tue | wed | thu | fri | sat | sun
TIME	now | CLOCK [DATE]
CLOCK	h:m | h:mm | hh:m | hh:mm
DATE	today | tomorrow | {MONTHS {d | dd} [,year]}
MONTHS	jan | feb | mar | apr | may | jun | jul | aug | sep | oct | nov | dec

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   [TOLERANCE=]number

Values   Any number of minutes starting from 0

Usage Notes   The following usage notes apply:

- If tolerance is not specified, then Oracle Reports Services reruns 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 Oracle Reports Services reuses 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   The following usage notes apply:

- 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   `[DATEFORMATMASK=]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   `[NUMBERFORMATMASK=]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
```
EXPRESS_SERVER="server=[server]/domain=[domain]/user=[userid]/
password=[passwd]"
```

Syntax with RAM

```
EXPRESS_SERVER="server=[server]/domain=[domain]/
user=[userid]/password=[passwd]/ramuser=[ramuserid]/
rampassword=[rampasswd]/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 user ID to log on to the Express Server.
password	is the password for the user ID.
ramuser	is the user ID to log into the RDBMS.
rampassword	is the password for the RDBMS.

`ramexpressid`	is the Oracle Sales Analyzer database user ID. 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 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:

Parameter	Description	Setting
sl	Server Login	-2: Host (Domain Login)
		-1: Host (Server Login)
		0: No authentication required
		1: Host (Domain Login) and Connect security
		2: Host (Domain Login) and Call security
		3: Host (Domain Login) and Packet security
		4: Host (Domain Login) and Integrity security
		5: Host (Domain Login) and Privacy security
		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.
st	Server Type	:1: Express Server
ct	Connection Type	0: Express connection
sv	Server Version	1: Express 6.2 or greater

Usage Notes   The following usage notes apply:

- You can have spaces in the string if necessary (for example, if the user ID 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 or 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 user name and password used to authenticate users to the restricted Oracle Reports Services. 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 Oracle Reports Services registered in WebDB. By default, the cookie expires after 30 minutes. When a cookie expires, subsequent requests (that is, ones sent to a secured Oracle Reports Services) 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 user name 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 file name or list of file names that contain a valid XML report definition, with path information prefixed to the file name or file names 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 with 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 file name.

B

Oracle Reports Services Configuration Parameters

This appendix contains a comprehensive list of Oracle Reports Services configuration parameters:

Parameter	Description
CACHEDIR	CACHEDIR is the cache for Oracle Reports Services. 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`. For example: `CACHEDIR="C:\ORACLE_HOME\Report60\cache"`
CACHESIZE	CACHESIZE is the size of the cache in megabytes. If you expect to store the output of many of your reports in Oracle Reports Services cache, then you might want to increase this setting. If you do not expect to store a lot of output in the cache and have limited system resources, then you might want to reduce it. Once the cache grows beyond the set size, Oracle Reports Services cleans up the cached files on a first in, first out basis. The default value is 50. **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 Cache size (MB) setting.

Parameter	Description
CLUSTERCONFIG	CLUSTERCONFIG is the configuration of slave servers to the master server. Clustering allows you to run reports on multiple Oracle Reports Services. 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 Oracle Reports Services Clusters" for detailed instructions.
ENGLIFE	ENGLIFE is the maximum number of reports that an engine runs before shutting itself down. Oracle Reports Services then brings up fresh engines for new requests. The default value is 50.
FAILNOTEFILE	FAILNOTEFILE is path and file name of the notification message template that is sent to specified email addresses for jobs that fail to run. For example: FAILNOTEFILE="C:\ORACLE_HOME\Report60\failnote.dat"
IDENTIFIER	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 Oracle 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.
INITENGINE	INITENGINE is the initial number of runtime engines started by Oracle Reports Services. The server process spawns this many engines when it is started. It waits two minutes for these engines to connect to it and shuts itself down if they fail to do so. If the engines cannot connect in this amount of time, then there is usually some setup problem. The default value is 1.
LOGOPTION	LOGOPTION is the type of log information you want inserted into the log file. The options are alljob, failedjob, and succeededjob. For example: LOGOPTION="alljob"

Parameter	Description
MAILPROFILE	If DESTYPE=MAIL, then Oracle Report Services sends your mail to a specific destination. MAILPROFILE allows you to specify the mail profile and password to be used when mailing reports from Oracle Report Services. For example: MAILPROFILE="mailprofileid/password" This parameter is only applicable for Windows NT. Windows NT has it's own Windows message system, and MS Exchange uses this system (specifically, MAPI). For MAPI to work, you need to provide a profile entry that corresponds to the entry created in MS Exchange so that MAPI knows the sender information. If you are using Netscape 4.7 or later, you do not need to setup the MAILPROFILE parameter. You do need to create entries in the Netscape phone book for all receivers.
MAXCONNECT	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). The default value is 20.
MAXENGINE	MAXENGINE is the maximum number of runtime engines available to Oracle Reports Services to run reports. The server process attempts to keep no more than this many engines active. Ensure you have sufficient memory and resources available to accommodate this number of engines. The default value is 1. **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.
MAXIDLE	MAXIDLE is the maximum amount of time an engine is allowed to be idle before being shut down. Oracle Reports Services does not shut down the engine if doing so would reduce the number of available engines to less than those defined in the MINENGINE. T default value is 30 minutes. **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 MAXIDLE time (minutes) before engine shutdown setting.

Parameter	Description
MINENGINE	MINENGINE is the minimum number of runtime engines Oracle Reports Services should have available to run reports. The server process attempts to keep at least this many engines active. Ensure that you have sufficient memory and resources available to accommodate this many engines. The default value is 0. **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.
PERSISTFILE	PERSISTFILE indicates the location of Oracle Reports Services `.DAT` file, which contains the details of scheduled jobs. If PERSISTFILE is not specified, then the default is `ORACLE_HOME\REPORT60\SERVER`. For example: PERSISTFILE="C:\ORACLE_HOME\Report60\repserver.dat"
REPOSITORYCONN	REPOSITORYCONN is the database connection string that connects Oracle Reports Services to the database when the server starts up. The database takes a snapshot of Oracle Reports Services queue activity (that is, scheduled jobs) whenever jobs are run. To create a queue activity table in your database, you must run `rw_server.sql` script. For example: REPOSITORYCONN="repserver_schema/password@mydb"
SECURITY	SECURITY is the security level (0, 1, 2, or 3) for accessing cached output files through the Oracle Reports Queue Manager. A 0 means that anyone can access a job's cached output. A 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. A 2 means that only the same process that sent the job can access the job's cached output. A 3 means that the cached output cannot be accessed. The default value is 1.

Parameter	Description
SECURITYTNSNAME	SECURITYTNSNAME is the TNS name of the Oracle WebDB database that is used for authenticating users to Oracle Reports Services. Oracle Reports Services uses Oracle WebDB to perform a security check and to ensure that users have access privileges to run the report to the restricted Oracle Reports Services and, if requested, output to a restricted printer.
	When the SECURITYTNSNAME parameter is set, you must add information about Oracle Reports Services, printers, and reports in WebDB to process report requests through Oracle Reports Services. For example:
	SECURITYTNSNAME="sec_db"
	See Chapter 5, "Controlling User Access to Reports" for more information.
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 Oracle Reports Services sharing the same ORACLE_HOME because each Oracle Reports Services can search different directories. For 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. For example:
	SUCCNOTEFILE="C:\ORACLE_HOME\REPORT60\succnote.dat"
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. For example
	TEMPDIR="C:\ORACLE_HOME\Report60\temp"

C

Environment Variables

This appendix contains detailed explanations of environment variables and configuration parameters that pertain to Oracle Reports Services. See the table below for a list of Web CGI and Servlet environments variables.

Environment variables are the configuration parameters used to control or customize the behavior of Oracle Reports Services. For Windows NT, environment variables are set using the Registry Editor. For UNIX, variables can be set using a shell script.

Variable	Description
REPORTS60_COOKIE_EXPIRE	Determines the expire time of the cookie in minutes. The default value is 30.
	Cookies save encrypted user names and passwords on the client-side when users log on to a secured Oracle Reports Services 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 Oracle Reports Services), users must re-authenticate to run the report.
REPORTS60_DB_AUTH	Specifies the database authentication template used to log on to the database. The default value is `dbauth.htm`.
REPORTS60_ENCRYPTION_KEY	Specifies the encryption key used to encrypt the user name and password for the cookie. The encryption key can be any character string. The default value is reports6.0.

Variable	Description
REPORTS60_CGIDIAGBODYTAGS	For the Oracle Reports Web CGI, specifies HTML tags that are inserted as a <BODY…> tag in the RWCGI60 diagnostic/debugging output. For example, you might want to use this environment to set up text and background color or image.
REPORTS60_CGIDIAGHEADTAGS	For the Oracle Reports Web CGI, specifies HTML tags to insert between <HEAD> …</HEAD> tags in the RWCGI60 diagnostic and debugging output. For example, you might want to use this environment to set up <TITLE> or <META…> tags.
REPORTS60_CGIHELP	For the Oracle Reports Web CGI, defines URL and URI of the RWCGI60 help file, which is navigated to when RWCGI60 is invoked with the empty request: `http://your_webserver/rwcgi60?`. For example., setting it to `http://www.yahoo.com` goes to that URL; setting it to `myhelpfile.htm` displays the file: `http://your_webserver/myhelpfile.htm` If this parameter is not defined, then a default help screen is displayed.
REPORTS60_CGIMAP	For the Oracle Reports Web CGI, defines fully qualified file name and location of the RWCGI60 map file if map file configuration is used. For example: `C:\ORANT\REPORT60\cgicmd.dat)`
REPORTS60_CGINODIAG	For the Oracle Reports Web CGI, when defined, disables all debugging and diagnostic output, such as help and showmap, from RWCGI60. For example, the following does not work when REPORTS60_CGINODIA is defined: `http://your_webserver/rwcgi60/help?`
REPORTS60_REPORTS_SERVER	Specifies the default Oracle Reports Services 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.
REPORTS60_SSLPORT	If you are using SSL and you want to use a port number other than 443, then you can use this variable to set a different port number.The default value is 443.

Variable	Description
REPORTS60_SYS_AUTH	Specifies the authentication template used to authenticate the user name and password when users run report request to a restricted Oracle Reports Services.

D

Database Connection Strings

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.

Database Connection String	Oracle Reports Services Response	User Action
No USERID specified	Returns the database authentication form.	Types the Oracle or placeholder user name and password.
Oracle `username@database`	Looks for the Oracle user name and database pair in the connection string table to get the password. If Oracle Reports Services finds the password, then the report is run.	None.
	If the password cannot be found, then Oracle Reports Services returns the database authentication form.	Types the database password.

Database Connection String	Oracle Reports Services Response	User Action
Oracle `username/password@database`	Accepts the connection string and runs the report.	None.
Oracle `username/password`	Uses the local database and runs the report.	None.
	If there is no local database, then Oracle Reports Services returns the database authentication form.	Types the Oracle database.
`<$username>@database`	Looks for the placeholder user name in the connection string table. If the user name cannot be found, then Oracle Reports Services returns the database authentication form.	Types the Oracle user name and password.
	If Oracle Reports Services can find the placeholder user name in the table, then it looks for the Oracle user name and database name pair in the table to get the password. If Oracle Reports Services finds the password, then the report is run.	None.
	If the password cannot be found in the table, then Oracle Reports Services returns the database authentication form.	Types the database password.

Database Connection String	Oracle Reports Services Response	User Action
`<$username>/password@database`	Looks for the placeholder user name in the connection string table. If the user name is found, then Oracle Reports Services runs the report.	None.
	If the placeholder user name cannot be found, then it returns the database authentication form. The user must authenticate to run the report.	Types the Oracle user name and password.

E

Migrating from Web Cartridge to Web CGI

This appendix contains step-by-step instructions on how to migrate from the Web cartridge to a Web CGI. For the purposes of this appendix, it is assumed that you have an existing Oracle Reports Server that is configured on the Oracle Application Server (OAS) using the Web Cartridge.

> **Note:** If you configured the Oracle Reports Server using the WebDB Listener, any CGI-enabled Web server, or any Java-aware servlet, then you have already configured the Oracle Reports Server Web CGI. Migration is not necessary.

E.1 Benefits of Migrating to Web CGI

CGI is a component of the Web HTTP protocol. It is a standard, platform-independent way to dynamically communicate with the Oracle Reports Server. Benefits include the following:

- Openness

  Most Web servers support CGI. It is the most common implementation.

- Easy implementation

  CGI is faster and easier to implement than the Web Cartridge.

E.2 Steps for Migrating to Web CGI

Migrating to Web CGI involves the following steps:

1. Installing the software.

2. Configuring OAS

3. Configuring the Web CGI

4. Setting environment variables (optional)

5. Renaming the map files (optional)

6. Running a report using the Web CGI URL

7. Updating the report links on your Web pages

These steps are performed on your OAS machine and assume that you have an existing Oracle Reports Server using the Web Cartridge.

E.2.1 Step 1. Installing the Software

You need to do the following to install the software:

1. Install the Oracle Reports Developer Thin Client, if you have not already done so.

2. Ensure that a TNSnames service entry exists for the Oracle Reports Server in the `tnsnames.ora` file that is used by the Web CGI to communicate with the Oracle Reports Server.

   ■ If OAS is installed on a different machine than the Oracle Reports Server, then check the `tnsnames.ora` file to ensure that a TNSnames service entry exists for the Oracle Reports Server in the OAS `ORACLE_HOME`. Refer to the Oracle Reports Server 6*i* *Publishing Reports* manual for more information.

   ■ If OAS release 4.0.8 is installed on the same machine as the Oracle Reports Server, but in a different Oracle home, then you will need to add a TNSnames service entry for the Oracle Reports Server in the `tnsnames.ora` file in the OAS `ORACLE_HOME`. Refer to the Oracle Reports Server 6*i* *Publishing Reports* manual for more information.

   ■ If OAS release 4.0 (previous to OAS 4.0.8) is installed on the same machine as the Oracle Reports Server and is in the same `ORACLE_HOME`, then no additional TNSnames entries are required.

E.2.2 Step 2. Configuring OAS

To configure OAS, you do the following:

1. Start your browser

2. Click **OAS Manager** in the Oracle Application Server Welcome page.

3. Click the + icon beside the Web home site icon in the OAS Manager navigational tree.

4. Expand the HTTP listener node.

5. Create a listener if necessary or expand the listener you want to use.

6. Click **Directory** and configure the OAS directory mapping using the information in the following table.

Directory Description	Physical Directory Example	Virtual Directory Example	Permissions Required
BIN	C:\OAS\BIN	/CGI-BIN	read and execute

E.2.3 Step 3. Configuring the Web CGI

To configure the Web CGI, you do the following:

1. On Windows, copy the `rwcgi60.exe` file (located in the `ORACLE_HOME\BIN` directory) to your CGI-BIN directory. On UNIX, copy the `rwcgi60` file (located in the `ORACLE_HOME\BIN` directory) to your CGI-BIN directory. The CGI-BIN directory is defined in the OAS directory mapping. In this example it is `C:\OAS\BIN`.

2. If OAS and Oracle Reports Developer are in different home directories, then you also need to copy the Oracle Reports Developer home into the OAS home.

On Windows, if the Oracle Reports Developer home is `D:\ORANT\REPORTS60` and the OAS home is `E:\ORANT\OAS`, then you need to create a REPORTS60 subdirectory in `E:\ORANT\OAS` and move the template files from `D:\ORANT\REPORTS60` to `E:\ORANT\OAS\REPORTS60`.

On UNIX, if the Oracle Reports Developer home is
`/private1/oracle6i/reports60` and OAS is `/private1/oas`, then you
would run the following commands:

```
cd/private1/oas
mkdir reports60
cd reports60
cp /private1/oracle6i/reports60/dbauth.htm
cp /private1/oracle6i/reports60/sysauth.htm
cp /private1/oracle6i/reports60/dbsysdif.htm
cp /private1/oracle6i/reports60/dvsyssam.htm
```

E.2.4 Step 4. Setting Environment Variables (Optional)

If you set any Web Cartridge environment variables (for example,
`REPORTS60_OWSHELP` to specify the location of the map or help file), then you
need to set environment variables for Web CGI (for example,
`REPORTS60_CGIHELP`).

To display the Web Cartridge environment variables that are currently in use, start
your browser and type your OAS Cartridge URL with the snow environment
variables command. For example, enter the following:

```
http://my_webserver/rwows?showenv
```

E.2.4.1 Windows NT

Before you begin, Oracle Corporation recommends that you back up the registry
before making any changes. Do the following steps:

1. Choose **Start**→**Run** on your desktop.

2. Type `regedit` to display the Registry Editor.

3. Expand **Hkey_Local_machine**→**Software**→**Oracle**.

4. Choose the **Edit**→**New**→**String** value to add the Web CGI environment
    variable. Refer to the Oracle Reports Server 6*i Publishing Reports* manual for a
    description of CGI-specific environment variables.

Refer to your operating system's documentation for more information.

E.2.4.2 UNIX

You might want to create a shell script that sets environment variables on your OAS machine. To do this, you create a file that contains the following command for each environment variable that you want set. For example:

```
setenv REPORTS60_CGIHELP myhelp.html
```

Refer to the Oracle Reports Server 6*i* *Publishing Reports* manual for a description of the CGI-specific environment variables.

E.2.5 Step 5. Renaming the Map Files (Optional)

If you use a key mapping file to simplify or hide parameters, then you will need to rename the key mapping file (for example, owscmd.dat) that was for the Web Cartridge to file name that Web CGI can recognize (for example, cgicmd.dat). You can copy and rename the owscmd.dat file (located in ORACLE_HOME\REPORTS60) to cgicmd.dat.

This completes the migration from Web Cartridge to Web CGI.

E.2.6 Step 6. Running a Report Using the Web CGI URL

You need to test that you have successfully migrated to Web CGI. You can test the configuration so ensure that it can communicate with the Oracle Reports Server. The URL for Web CGI is different than the URL for the Web Cartridge. Run a test using the following Web CGI URL. For this example, it is assumed that the REPORTS60_REPORT_SERVER environment variable was set to point to a default Oracle Reports Server. The SERVER command line argument is not needed in this case.

```
http://your_webserver/CGI-BIN/RWCGI60.EXE?REPORT=your_report.RDF+userid=username/password@
mydb+DESFORMAT=HTML+DESTYPE=CACHE
```

Notice that instead of using the RWOWS60 executable (for Web Cartridge) you are calling the RWCGI60 executable (for Web CGI) from the CGI-BIN path (that was defined in Step 2) to call the URL. The arguments that follow the ? (question mark) are the same, regardless of whether you are using Web Cartridge or Web CGI to communicate with the Oracle Reports Server.

If you cannot successfully run the report, then refer to the Oracle Reports Server 6*i* *Publishing Reports* manual for troubleshooting assistance.

E.2.7 Updating the Report Links on Your Web Page

If you maintain a Web page with links to run report requests, then you will need to change the URL reference to call the RWCGI60 executable from the CGI-BIN path.

If you configured your Oracle Reports Server for access control using Oracle WebDB, then you will need to change the Web gateway value in the Oracle Reports Server access control, which was created for the Oracle Reports Server.

F

Troubleshooting

This appendix contains information on how to troubleshoot your Oracle Reports Services configuration.

Problem Description	Probable Cause and Solution
Oracle Reports Services appears to hang when you start it.	You might have made a syntactical error in the `tnsnames.ora` file and Oracle Reports Services cannot resolve the TNS name.
	Alternatively, you could try rebooting in case the cause is a memory problem.
You get the error "Daemon failed to listen to port."	If you start up an Oracle Reports Services that is listening to the same port as an already running Oracle Reports Services, then you receive this error. It could also be a problem with your Net8 or TCP/IP setup.
You get an error about being unable to initialize the printer (REP-3002).	Ensure Oracle Reports Services has access to printers. For Windows NT, the System Account does not usually have access to printers. It could be that you installed Oracle Reports Services 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.

Problem Description	Probable Cause and Solution
Upon starting Oracle Reports Services, you get server specific error 186.	Typically this indicates a problem in `tnsnames.ora` or `sqlnet.ora`. Check the entry for Oracle Reports Services 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 Oracle Reports Services.
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, then do not 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.

Problem Description	Probable Cause and Solution
Problems running Oracle Reports Services as a Windows NT Service.	If you install Oracle Reports Services service to run under a user other than SYSTEM, then ensure the user account:
	■ Has the Password Never Expires option selected in the User Manager.
	■ Has membership in the appropriate groups to run Oracle Reports Services and access the report files.
	■ Has at least print permission to a default printer.
	■ 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, then click the **Add**.
	When starting the service, you might need to explicitly specify the domain as well as the user name (user name and 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 Oracle Report Builder online help.
ops$ account is not working.	For security reasons, ops$ accounts are not supported by Oracle Reports Services. If you pass a command line with USERID=/ to Oracle Reports Services, then an error is generated because it tries to use the user name of Oracle Reports Services process rather than the user name of the client.
Database roles not working as expected.	If you are using database roles, then Oracle Reports Services gets and then sets the default roles for the job request's database connection. If the default roles require a password, then Oracle Reports Services logs 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 Oracle Reports Services gets and then sets the default roles on a per job basis, you cannot share roles between jobs. This is done to preserve security.

Problem Description	Probable Cause and Solution
URL mapping is not working.	Ensure you have a valid key mapping file. It must be named `cgicmd.dat` (for the Oracle Reports Web CGI or Servlet) in the `REPORT60` directory, or named according to the value set in the REPORTS60_CGIMAP environment variable.
	To ensure the key mapping file can be found, first try the following (a Web CGI example) and verify that your key entry has been correctly parsed in the resulting page:
	http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe/showmap?
	Then try, running the report using the key map entry, where `your_key` is a valid key entry in the key mapping file:
	http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe?your_key
Cannot shutdown the queue from the Oracle Reports Queue Manager.	You should not leave the user name 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 Oracle Reports Queue Manager (**Options→Privileges→Administrator**), you can specify any user name and password. The user name and password that you specify the first time are the administrator's until you change it.
Cannot run Oracle Reports Services as an NT Service under LocalSystem.	If Oracle Reports Services is to be run as an NT service under the LocalSystem user ID, then the system administrator must ensure that the following line is in the `sqlnet.ora` file, otherwise the server cannot be accessed:
	`sqlnet.authentication_services=(NONE)`
Problems finding files.	Since network drives are mapped to a drive letter on a per user basis, these mappings are no longer in effect when the Windows NT user logs off. Oracle Reports Services 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 Oracle Reports Services parameters, Web CGI and Servlet command mappings, and each hard-coded path name in each report being run.

Problem Description	Probable Cause and Solution
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, then 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 Oracle Reports Services report definition file in WebDB.	In WebDB, verify the access controls that you defined for the printer, Oracle Reports Services, 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, then it is possible that you incorrectly entered the OS Printer name. ■ Access to Oracle Reports Services 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 Oracle Reports Services.

Problem Description	Probable Cause and Solution
Reports are not running when the URL is requested.	Check for the following:

Check for the following:

- Ensure the Web server is responding (for example, by trying to bring up your Web server administration page). Refer to your Web server installation documentation.

- Ensure your Web CGI or Servlet executable has been found and is responding. For Windows 95 and Windows 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/rwcgi60 or
  http://your_webserver/rwows

  A help page should appear. If it does not, then 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.

- Ensure that the REPORTS60_CGINODIAG (for Web CGI or Servlet) environment variable is not defined, otherwise all diagnostic output is disabled. Test this by typing one of the following:

  http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe/ showenv?

  http://your_webserver/rwows/showenv?

  This also allows you to view the other parameters or environment variables.

Problem Description	Probable Cause and Solution
	■ Ensure 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_re pserver+report=your_report.rdf+ userid=scott/tiger@mydb+desformat=html
	http://your_webserver/rwows?server=your_repserver+ report=your_report.rdf+userid=scott/tiger@my_db+ desformat=html
	If the report does not display, then check to ensure that:
	■ `Your_report.rdf` runs correctly from Oracle Report Builder or Oracle Reports Runtime `Your_report.rdf` is located in a directory specified under `REPORTS60_PATH`.
	■ The database connection string is correct.
	■ The Oracle Reports Server you are trying to run your report to might be restricted. If so, then you need to be given access privileges to the server. Contact your Oracle Reports Services system administrator.
	■ The report you are trying to run might be restricted. If so, then you need to be given access privileges to run it to a restricted Oracle Reports Server. Contact your Oracle Reports Services system administrator.
	Remember that the Oracle Reports Server must have access to the report and any external files used by the report.
	When sending a report to the Oracle 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 Oracle 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 fails.
Report does not output to the printer.	You might have access privileges to run a report to restricted Oracle Reports Server, but might not have access privileges to the printer you are trying to output to. Contact the Oracle Reports Services system administrator.

Problem Description	Probable Cause and Solution
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, then contact your system administrator.
	If you trying to run your report to a restricted Oracle Reports Server, then the Web Gateway URL defined in the Server Access in WebDB might be incorrect.
	In WebDB, click **Administrator** from the WebDB main menu. Then, click **Oracle Reports Developer Security** and **Server Access**. Search for the Oracle 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 Oracle Reports Services system administrator privileges can access Oracle Reports Services Security in WebDB.

Glossary

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.

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 Oracle Reports Services, they must authenticate. If they successfully log on, then their browser is sent an encrypted cookie. When a cookie has expired, subsequent requests (that is, ones that sent to a secured Oracle Reports Services) 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 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, or later, and Internet Explorer 4.0, or later, 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 is delivered, how long it takes, or if multiple packets arrive in the order they were sent. Protocols built on top of this add the notions of connection and reliability.

Net8

This is the Oracle 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.

Oracle Internet Application Server

Oracle Internet 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 Internet 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 user name

A placeholder user name enables users to log on to the database using their personal user name rather than the Oracle database user name (for example, `$user_name@database`). A placeholder user name 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 might receive East coast sales results and another might receive West coast sales results).

The first time users log on to the database, however, they must log on using the Oracle user name and password. For subsequent requests, Oracle Reports Services looks for the user's personal user name in the database connection table. If it is found, then Oracle Reports Services 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. Oracle Reports Services can push reports to WebDB site by scheduling the report request to run automatically on a secured Oracle Reports Services. The end user clicks the link on the WebDB site to view the report.

Oracle Reports Queue Manager

Enables you to monitor and manipulate job requests that have been sent to Oracle Reports Services.

Oracle Reports Launcher

An application that utilizes the functionality provided by the Oracle Reports ActiveX control, such submitting a request to run the specified report to Oracle Reports Services.

Oracle Reports Services

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. Oracle Reports Services handles client requests to run reports by entering all requests into a job queue.

Oracle Reports Servlet

An interface between a Java-based Web server and Oracle Reports Runtime, enabling you to run report dynamically from your Web browser.

Oracle Reports Web CGI

An interface between a CGI-aware Web server and Oracle 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 Oracle Reports Services (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

A Net8 file that contains connect descriptions mapped to service names. The file can 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 through the Internet. It is also the text string format clients use to encode requests to Oracle Internet 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 Services uses WebDB to control end user access to reports published on the Web by storing information about report requests, the secured server, and any Oracle Reports Services 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 Oracle Reports Services report definition file package). Running 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.

Index

Symbols

A