

Oracle[®] Reports Services

Publishing Reports to the Web with Oracle9i Application Server

Release 1.0.2 for Windows NT and UNIX

Part No. A86784-02

November 2000

ORACLE[®]

Publishing Reports to the Web with Oracle9i Application Server, Release 1.0.2

Part No. A86784-02

Copyright © 1996, 2000, Oracle Corporation. All rights reserved.

Primary Author: Frank Rovitto

Contributing Author: Pat Hinkley

Contributors: Chan Fonseka, Shaun Lin, Paul Narth, Padma Hariharan, Ashok Natesan, Danny Richardson, Ravikumar Venkatesan, Viswanath Dhulipala, and Jeff Tang

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

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

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark, and Oracle Developer, Oracle Reports, Oracle9i Application Server, Express, Oracle Report Services, Oracle9i Developer Suite, Oracle Universal Installer, and Oracle Installer are trademarks or registered trademarks of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Portions copyright © Blue Sky Software Corporation. All rights reserved. All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

Contents

Send Us Your Comments	xi
Preface	xiii
Oracle Reports Services New Features and Changes.....	xiii
Intended Audience	xiv
Structure.....	xiv
Related Documents.....	xiv
Notational Conventions.....	xv
Part I Publishing Reports	
1 Publishing Architecture and Concepts	
1.1 Oracle Reports Services.....	1-1
1.2 Oracle Reports Services Architecture.....	1-2
1.2.1 Web Architecture: Server Configurations	1-3
1.2.1.1 Processing Web Reports.....	1-4
1.2.2 Non-Web Architecture: Server Configuration.....	1-6
1.2.2.1 Processing Reports.....	1-6
1.3 Oracle Reports Services Configuration Choices	1-7
1.3.1 Enable Web and Non-Web Requests.....	1-7
1.3.2 Choose the Oracle Reports Services Server CGI or Servlet.....	1-8
1.3.3 Choose the Location of Oracle Reports Services	1-8

2 Installing Oracle9i Application Server with Oracle Reports Services

2.1	About the Oracle Universal Installer	2-1
2.2	About the Oracle HTTP Server <i>powered by Apache</i>	2-2

3 Configuring the Oracle Reports Services Server on Windows NT and UNIX

3.1	Starting and Stopping the Oracle Reports Services Server	3-1
3.1.1	Starting the Oracle Reports Services Server	3-1
3.1.1.1	Starting the Oracle Reports Services Server on Windows NT	3-2
3.1.1.2	Starting the Oracle Reports Services Server on UNIX	3-2
3.1.2	Starting the Oracle Reports Services Server on Windows NT as a Non-Service ..	3-3
3.2	Stopping the Oracle Reports Services Server	3-3
3.2.1	Stopping or Deinstalling the Oracle Reports Services Server on Windows NT ...	3-4
3.2.2	Stopping the Oracle Reports Services Server on UNIX	3-4
3.3	Configuring the Oracle Reports Services Server Servlet	3-4
3.3.1	Configuring the Oracle Reports Services Server Servlet with JSDK	3-5
3.3.2	Configuring the Oracle Reports Services Server Servlet with JServ	3-7
3.4	Configuring the Oracle HTTP Server <i>powered by Apache</i> Listener	3-8
3.5	Configuring the Web Server	3-9
3.5.1	Configuring the Oracle Reports Services Server CGI	3-10
3.5.1.1	Configuring the Oracle Reports Services Server CGI	3-10
3.5.1.2	Creating a Service Entry for the Oracle Reports Services Server	3-11
3.5.1.3	Setting the Default Oracle Reports Services Server (Optional)	3-11
3.5.1.3.1	Windows NT	3-12
3.5.1.3.2	UNIX	3-12
3.6	Configuring the Oracle Reports Services Server with Environment Variables	3-12
3.6.1	Configuring the Oracle Reports Services Server in Windows NT with Environment Variables	3-12
3.6.1.1	Setting the Environment Variables (Optional)	3-13
3.6.1.2	Starting the Oracle Reports Services Server	3-13
3.6.2	Configuring the Oracle Reports Services Server on UNIX with Environment Variables	3-14
3.6.2.1	Setting the Environment Variables (Optional)	3-14
3.6.2.2	Starting the Oracle Reports Services Server on UNIX	3-15
3.7	Environment Variables	3-15

3.8	Running a Report Request from a Web Browser.....	3-16
3.8.1	Other Steps.....	3-17
3.9	Modifying the Oracle Reports Services Server Configuration (Optional).....	3-17
3.9.1	Updating the Database with Job Queue Activity.....	3-18
3.9.1.1	On the Oracle Reports Services Server Machine.....	3-18

4 Running Report Requests

4.1	Report Request Methods.....	4-1
4.2	Duplicate Job Detection.....	4-3
4.2.1	Usage Notes.....	4-3
4.3	Using a Key Map File.....	4-4
4.3.1	Enabling Key Mapping.....	4-5
4.3.2	Mapping URL Parameters.....	4-6
4.4	Specifying Report Requests.....	4-6
4.4.1	Building a Report.....	4-7
4.4.2	Specifying a Report Request from a Web Browser.....	4-8
4.4.3	Scheduling Reports to Run Automatically.....	4-9

5 Oracle Reports Services Security with Oracle Portal

5.1	Overview.....	5-2
5.1.1	Creating a Security DLL for Oracle Reports Services 6i Security in a Windows Environment.....	5-3
5.1.2	Creating a Security Library for Oracle Reports Services 6i Security in a UNIX Environment.....	5-4
5.2	Database-Level Security.....	5-4
5.3	Application-Level Security.....	5-7
5.4	Integration with Oracle Portal.....	5-8
5.4.1	Sharing Authentication Information Between Oracle Portal and Oracle Reports Services Servers.....	5-9
5.5	Oracle Portal Integration Architecture.....	5-9
5.6	Installing Oracle Reports Services Security in Oracle Portal.....	5-10
5.6.1	Step 1: Installing Oracle Portal Into an Oracle Database.....	5-10
5.6.2	Step 2: Installing Oracle Reports Services.....	5-10

5.7	Configuring the Security Environment.....	5-11
5.7.1	Step 1: Enabling Oracle Reports Services Security within Oracle Portal	5-11
5.7.1.1	RW_ADMINISTRATOR	5-12
5.7.1.2	RW_DEVELOPER.....	5-12
5.7.1.3	RW_POWER_USER.....	5-12
5.7.1.4	RW_BASIC_USER.....	5-12
5.7.2	Step 2: Adding SECURITYTNSNAMES and PORTALUSERID Parameters	5-13
5.7.3	Step 3: Starting Oracle Portal.....	5-14
5.8	Printer Access.....	5-14
5.9	Creation of an Oracle Portal Content Area.....	5-15
5.10	Setting Up and Deploying a Report.....	5-16
5.11	Creating and Enabling an Oracle Portal User to Administer Security	5-16
5.11.1	Creating and Enabling User REPORTSDEV to Administer Security	5-17
5.12	Setting Up Access Controls in Oracle Portal	5-22
5.13	Registering a Report.....	5-22
5.13.1	Registering a Server	5-23
5.13.2	Creating Report Definition File Access	5-27
5.14	Deploying a Report	5-38
5.14.1	Deploying a Report to an Oracle Portal Content Area	5-39
5.15	Running a Report.....	5-46
5.16	Publishing Report Outside of Oracle Portal	5-49

6 Configuring Oracle Reports Services Server Clusters

6.1	Clustering Overview.....	6-2
6.2	Configuring Oracle Reports Services Servers in a Cluster Example.....	6-3
6.2.1	Enabling Communication Between Master and Slaves.....	6-4
6.2.2	Configuring the Master Server.....	6-5
6.2.3	Running Reports in a Clustered Configuration.....	6-7
6.2.4	Resubmitting Jobs When an Engine Goes Down	6-7
6.2.5	Adding Another Slave Server to the Master	6-8

7 Customizing Reports at Runtime

7.1	Overview	7-2
7.1.1	Creating and Using XML Report Definitions	7-2
7.2	Creating an XML Report Definition	7-3
7.2.1	Required Tags	7-4
7.2.2	Partial Report Definitions	7-5
7.2.2.1	Formatting Modifications Example.....	7-7
7.2.2.2	Formatting Exception Example.....	7-9
7.2.2.3	Program Unit and Hyperlink Example	7-10
7.2.2.4	Data Model and Formatting Modifications Example	7-11
7.2.3	Full Report Definitions	7-12
7.3	Running XML Report Definitions.....	7-19
7.3.1	Applying an XML Report Definition at Runtime.....	7-20
7.3.1.1	Applying One XML Report Definition	7-20
7.3.1.2	Applying Multiple XML Report Definitions	7-20
7.3.1.3	Applying an XML Report Definition in PL/SQL.....	7-21
7.3.1.3.1	Applying an XML Definition Stored in a File.....	7-21
7.3.1.3.2	Applying an XML Definition Stored in Memory	7-21
7.3.2	Running an XML Report Definition by Itself.....	7-25
7.3.3	Performing Batch Modifications	7-25
7.4	Debugging XML Report Definitions	7-26
7.4.1	XML Parser Error Messages	7-26
7.4.2	Tracing Options	7-26
7.4.3	RWBLD60	7-28
7.4.4	TEXT_IO	7-29
7.5	XML Tag Reference	7-30
7.5.1	<!-- comments -->	7-30
7.5.2	<![CDATA[]]>.....	7-31
7.5.3	<condition>	7-32
7.5.4	<customize>.....	7-34
7.5.5	<data>	7-36
7.5.6	<dataSource>	7-37
7.5.7	<exception>.....	7-39
7.5.8	<field>.....	7-41
7.5.9	<formLike>	7-46

7.5.10	<formula>.....	7-47
7.5.11	<function>	7-50
7.5.12	<group>	7-51
7.5.13	<groupAbove>.....	7-53
7.5.14	<groupLeft>	7-54
7.5.15	<labelAttribute>	7-56
7.5.16	<layout>.....	7-58
7.5.17	<link>	7-61
7.5.18	<matrix>	7-64
7.5.19	<matrixCell>	7-67
7.5.20	<matrixCol>	7-68
7.5.21	<matrixRow>	7-69
7.5.22	<object>.....	7-70
7.5.23	<programUnits>	7-72
7.5.24	<properties>	7-73
7.5.25	<property>	7-75
7.5.26	<report>	7-78
7.5.27	<section>.....	7-80
7.5.28	<select>	7-82
7.5.29	<summary>.....	7-83
7.5.30	<tabular>	7-88

Part II Appendixes

A Controlling User Access to Reports by Defining Calendars

A.1	Creating Availability Calendars.....	A-1
A.2	Availability Calendar Example	A-2
A.2.1	Creating a Daily Calendar.....	A-2
A.2.2	Creating the Maintenance Calendar	A-3
A.2.3	Creating the Christmas Calendar.....	A-4
A.2.4	Creating a Combined Availability Calendar.....	A-5

B RWCLI60 Command Line Arguments

B.1	Syntax.....	B-1
B.2	Usage Notes.....	B-1

C Oracle Reports Services Configuration Parameters

D Environment Variables

E Database Connection Strings

F Migrating from Web Cartridge to CGI

F.1	Benefits of Migrating to CGI.....	F-1
F.2	Steps for Migrating to CGI.....	F-2
F.2.1	Step 1. Installing the Software.....	F-2
F.2.2	Step 2. Configuring OAS.....	F-3
F.2.3	Step 3. Configuring the CGI	F-3
F.2.4	Step 4. Setting Environment Variables (Optional)	F-4
F.2.4.1	Windows NT.....	F-4
F.2.4.2	UNIX.....	F-5
F.2.5	Step 5. Renaming the Map Files (Optional).....	F-5
F.2.6	Step 6. Running a Report Using the CGI URL.....	F-5
F.2.7	Updating the Report Links on Your Web Page	F-6

G Troubleshooting

Glossary

Index

Send Us Your Comments

Publishing Reports to the Web with Oracle9i Application Server, Release 1.0.2

Part No. A86784-02

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 and Changes

The following features are new for this release:

New Features and Changes	See
HTTP <i>powered by Apache</i>. The <code>httpds.conf</code> file has changed to <code>httpd.conf</code> .	Chapter 3, "Configuring the Oracle Reports Services Server on Windows NT and UNIX"
Oracle Portal. Oracle Reports Services 6i security with Oracle Portal.	Chapter 5, "Oracle Reports Services Security with Oracle Portal"

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 Services server.
- Chapter 4** Describes the various methods for running reports to the Oracle Reports Services server.
- Chapter 5** Describes how the Oracle Reports Services can be integrated with Oracle Portal 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 Portal, or the Oracle Report Services, refer to the following manuals:

- *Oracle Reports Developer Building Reports*, A73172-01
- *Oracle Reports Developer Getting Started for Windows*, A73156-02
- *Oracle Portal 3.0: Tutorial*, A86188-01
- *Deploying Forms to the Web with Oracle9i 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.
<i>italicized text</i>	Used to introduce new terms.
<code>courier font</code>	Used for path and file names, and for code and text that you type.
<code>COURIER CAPS</code>	Used for file extensions (.PLL or .FMX) and SQL commands
<code>CAPS</code>	Used for environment variables, built-ins and package names, and executable names

Part I

Publishing Reports

Chapter 1, "Publishing Architecture and Concepts"

Chapter 2, "Installing Oracle9i Application Server with Oracle Reports Services"

Chapter 3, "Configuring the Oracle Reports Services Server on Windows NT and UNIX"

Chapter 4, "Running Report Requests"

Chapter 5, "Oracle Reports Services Security with Oracle Portal"

Chapter 6, "Configuring Oracle Reports Services Server Clusters"

Chapter 7, "Customizing Reports at Runtime"

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 Services server CGI or Oracle Reports Services 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 Services Queue Manager (RWRQM60) or through the API. The Oracle Reports Services 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 Services 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 tiers¹:

- 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.

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.

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.

Figure 1–1 Web Architecture, Three Machine Configuration

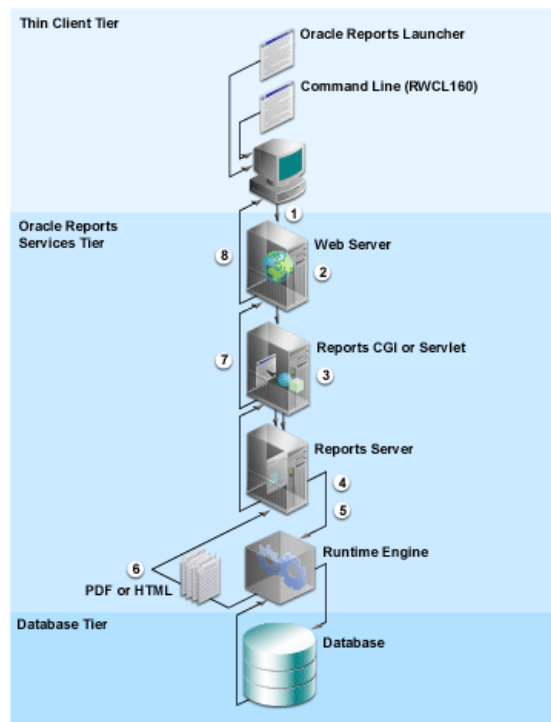
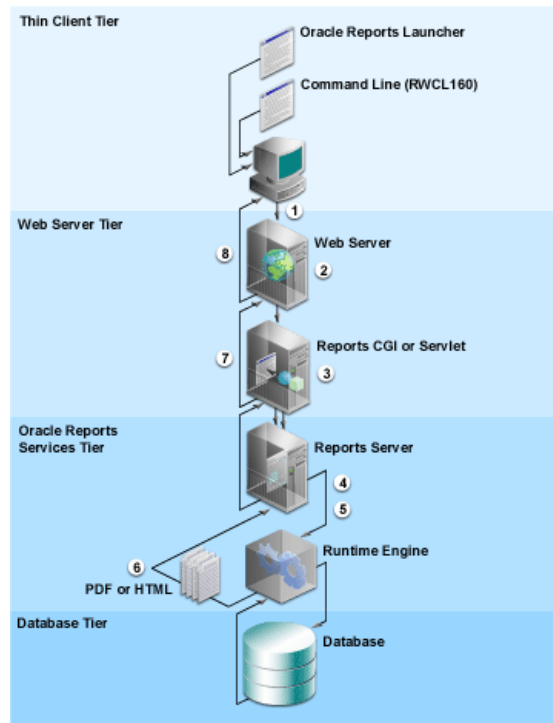


Figure 1–2 Web Architecture, Four Machine Configuration

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 Services server CGI or Oracle Reports Services server servlet, depending upon which one you have configured.

3. The Oracle Reports Services server CGI or servlet parses the request. If necessary, users are prompted to log on. The Oracle Reports Services 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 server.
4. If the request includes a time tolerance², 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,³ it sends the command line to that runtime engine for execution.
6. The runtime engine runs the report.
7. The Oracle Reports Services server 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.

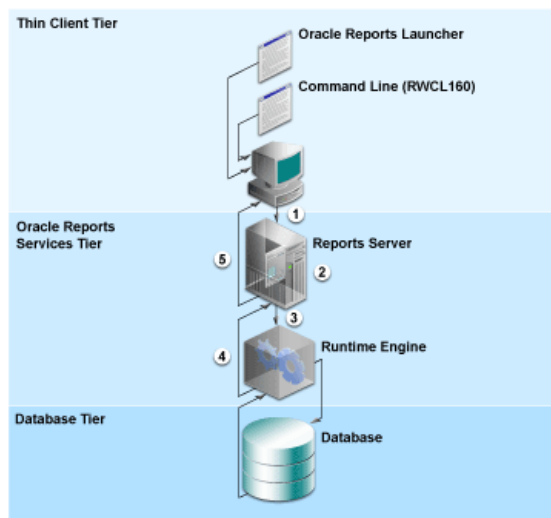
² For any job request that you send to the Oracle Reports Services, you can include a TOLERANCE argument. TOLERANCE defines the oldest output that the requester would consider acceptable. For example, if the requester specified five minutes as the TOLERANCE, Oracle Reports Services would check its cache for duplicate report output that had been generated within the last five minutes.

³ 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.

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 Services 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.

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 Services Queue Manager, or the Oracle Reports Services 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 (DESTTYPE) 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 Services Thin Client, which is comprised of the Oracle Reports Services Launcher, the Oracle Reports Services Queue Manager, and RWCLI60).

To enable users to launch reports from a Web client, you need to install either the Oracle Reports Services server CGI or servlet with your Web server to communicate between the Web server and Oracle Reports Services. The 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 Server 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 server 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 Server 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 server 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, Oracle9i Application Server or Oracle Portal Listener), then choose the Oracle Reports Services server CGI.
- If you are using a Java-based Web server, then choose the Oracle Reports Services server 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 Services and the Web server on the same machine reduces network traffic. The Oracle Reports Services server 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 server CGI or 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 Services Server on Windows NT and UNIX"](#) provides guidelines for configuring Oracle Reports Services using the Oracle Reports Services server servlet.

Installing Oracle9i Application Server with Oracle Reports Services

The Oracle Reports Services is installed as part of the Enterprise Edition of Oracle9i Application Server. 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 Oracle9i Application Server.

For more detailed information about installing Oracle Reports Services, refer to the *Oracle9i Application Server Installation Guide*. All necessary requirements and tasks are documented in this manual.

2.1 About the Oracle Universal Installer

The Oracle9i 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 Oracle9i 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 transactions² by leveraging the functionality supplied by Apache JServ (mod_jserv), which is described in the *Oracle9i Applications Server Overview Guide*.

Additional information about the Oracle HTTP Server *powered by Apache* can also be found in the *Oracle9i Application Server Installation Guide* and the *Apache Web Server, Release 1.3* manual on your CD-ROM.

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 about 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.

Configuring the Oracle Reports Services Server on Windows NT and UNIX

When you install the Oracle9i Application Server with the Oracle HTTP Server *powered by Apache*, the Oracle Reports Services server servlet and the Oracle Reports Services 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 Services server and the configuration environment variables.

3.1 Starting and Stopping the Oracle Reports Services Server


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

3.1.1 Starting the Oracle Reports Services Server

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

3.1.1.1 Starting the Oracle Reports Services Server on Windows NT

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

1. On the Oracle Reports Services 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 Services 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 Services 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 Services server has started. If your Oracle Reports Services server cannot start, then refer to [Appendix G, "Troubleshooting"](#) for more information.

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

3.1.1.2 Starting the Oracle Reports Services Server on UNIX

Do the following steps to start the Oracle Reports Services server on UNIX:

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

```
rwmts60 name=repserver
```


Run the following command to run the Oracle Reports Services server in the background:

```
rwmts60 name=repserver &
```

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

```
rwrqv60 server=repserver
```

Status columns (for example, NAME, OWNER, and DEST) for the Oracle Reports Services 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 Services 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 Services Server

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

3.2.1 Stopping or Deinstalling the Oracle Reports Services Server on Windows NT

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

1. On the Oracle Reports Services 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 Services Server on UNIX

Do one of the following to stop the Oracle Reports Services server:

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

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

- Oracle Reports Services server servlet with JSDK
- Oracle Reports Services server servlet with JServ

3.3.1 Configuring the Oracle Reports Services Server Servlet with JSDK

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

```
/private1/ias
```

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

```
/private1/ias/6iserver
```

You do the following steps to configure the Oracle Reports Services server servlet with JSDK:

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

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

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

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

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 Services server.

6. Start the Oracle Reports Services server servlet runner by running the following command:

```
servletrunner &
```

7. Verify that the Oracle Reports Services server 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 Services 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 Services server to acknowledge the changes.

3.3.2 Configuring the Oracle Reports Services Server Servlet with JServ

You do the following to configure the Oracle HTTP Server *powered by Apache* to run the Oracle Reports Services server 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 Oracle9i Application Server:

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.rwsgi.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/rwsgi
```

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

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

```
httpdctl start
```

8. Verify the Oracle Reports Services 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 Services 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=ConnectionString+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 Services 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 Services server CGI:

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

```
ScriptAlias /cgi-bin/      "/privatel/ias/6iserver/bin"
```
2. Start the Oracle Reports Services server.
3. Start the Oracle HTTP Server *powered by Apache* listener using the following command:

```
httpdctl start
```

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

The 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 CGI-BIN Physical and Virtual Directories

Directory Description	Physical Directory example	Virtual Directory Example	Permissions Required
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 Services Server CGI

The following steps are performed on the Web server machine.

3.5.1.1 Configuring the Oracle Reports Services Server CGI

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

In [Table 3–1, "CGI-BIN Physical and Virtual Directories"](#) the 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 Services server CGI must be in a path mapped as a CGI directory. The Oracle Reports Services RDF files must be in a path only accessible to the Oracle Reports Services server. If you choose the default installation of Oracle Reports Services 6i server and the Oracle Portal 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 Services Server

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

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

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

where:

<code>repserver.world</code>	is the name of the server instance and <code>.world</code> is the domain specified in the <code>NAMES.DEFAULT_DOMAIN</code> setting in the <code>sqlnet.ora</code> file. If the <code>NAMES.DEFAULT_DOMAIN</code> setting is not defined in the <code>sqlnet.ora</code> , then omit <code>.world</code> from the name of the server instance.
<code>repserver_machine.mydomain</code>	is the host name or IP address of the machine.
<code>1949</code>	is the port number to which the server is listening.

3.5.1.3 Setting the Default Oracle Reports Services Server (Optional)

You can, optionally, set defaults for the Oracle Reports Services 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 Services server that you are configuring (the TNSnames service entry name of the Oracle Reports Services server).

3.5.1.3.2 UNIX For UNIX, set the `REPORTS60_REPORTS_SERVER` environment variable to the name of the Oracle Reports Services 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 Services server that you are configuring (the TNSnames service entry name of the Oracle Reports Services server):

```
setenv REPORTS60_REPORTS_SERVER repserver
```

3.6 Configuring the Oracle Reports Services Server with Environment Variables

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

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

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

1. Setting the environment variables (optional)
2. Starting the Oracle Reports Services server

3.6.1.1 Setting the Environment Variables (Optional)

You can set two optional environment variables. The first lets the Oracle Reports Services 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 `tnsnames.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 Services server is installed, you can set the source path in the Oracle Reports Services server configuration file. See the `SOURCEDIR` parameter in [Appendix C, "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 Services Server

To start the Oracle Reports Services server, you do the following:

1. On the Oracle Reports Services 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 Services 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 Services 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 Services server has started. If your Oracle Reports Services server cannot start, then refer to [Appendix G, "Troubleshooting"](#) for more information.

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

3.6.2 Configuring the Oracle Reports Services Server on UNIX with Environment Variables

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

1. Setting environment variables (optional)
2. Starting the Oracle Reports Services 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 Services 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 Services server is installed, you can set the source path by using the `SOURCEDIR` parameter. See [Appendix C, "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 Services Server on UNIX

Do the following steps to start the Oracle Reports Services server on UNIX:

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

```
rwmts60 name=repserver
```

Run the following command to run the Oracle Reports Services server in the background:

```
rwmts60 name=repserver &
```

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

```
rwrqv60 server=repserver
```

Status columns (for example, NAME, OWNER, and DEST) for the Oracle Reports Services 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 Services server. Variables can be set using a command line for Windows NT and a shell script for UNIX.

Variable	Description
REPORTS60_COOKIE_EXPIRE	<p>Determines the expire time of the cookie in minutes. The default value is 30.</p> <p>Cookies save encrypted user names and passwords on the client-side when users log on to a secured Oracle Reports Services 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 Services server), user must re-authenticate to run the report.</p>

Variable	Description
REPORTS60_DB_AUTH	Specifies the database authentication template used to log on to the database. The default value is <code>dbauth.htm</code> .
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 <code>reports6.0</code> .
REPORTS60_REPORTS_SERVER	Specifies the default Oracle Reports Services server for Web requests. When this parameter is set, you can omit the <code>SERVER</code> command line argument in report requests to process them using the default server, or you can include the <code>SERVER</code> 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 Services server. The default value is <code>sysauth.htm</code> .

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 Services 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 Services server (Section 3.5.1.2, "Creating a Service Entry for the Oracle Reports Services 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 G, "Troubleshooting"](#) for more information.

2. View the status of the request (optional):
 - For Windows NT, start the Oracle Reports Services Queue Manager, choose to view the `repserver` queue. See the Oracle Reports Services 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 Services 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 Services Server Configuration (Optional)

When you start the Oracle Reports Services server for the first time, the Oracle Reports Services server is set with default configuration settings (for example, maximum and minimum engines). At some point, you might want to modify the Oracle Reports Services 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, "Oracle Reports Services Security with Oracle Portal"](#) for more information

- To configure the Oracle Reports Services server for load balancing, refer to [Chapter 6, "Configuring Oracle Reports Services Server Clusters"](#) for more information.
- To modify a report at runtime based on the audience, refer to [Chapter 7, "Customizing Reports at Runtime"](#) for more information.
- To modify the Oracle Reports Services server configuration file, see [Appendix C, "Oracle Reports Services Configuration Parameters"](#) for more information about the available configuration parameters.
- Modify environment variables, refer to [Appendix D, "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 Services server queue activity whenever jobs are run. When you start the Oracle Reports Services server, a connection to the database is made. By default, the Oracle Reports Services 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 Services 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 Services server for the changes to take effect.

3.9.1.1 On the Oracle Reports Services Server Machine

To update the database with job queue activity on the Oracle Reports Services 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 Services 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 Services servers, then it is recommended that you create a different schema in your database for each Oracle Reports Services server that requires snapshots. This prevents you from losing queue activity data when you restart the Oracle Reports Services server.

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

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

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 server. It uses a command line similar to the Oracle Reports Services 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 B, "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. The 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 Oracle Portal component enables you to add a link as an Oracle Portal component to an Oracle Portal site. This link points to a packaged procedure that contains information about the report request. Oracle Reports Services system administrators use Oracle Portal wizards to create the packaged procedure making it more convenient and secure to publish the report via the Web. Authorized users accessing the Oracle Portal site simply click the link to run the report. System administrators can run the report directly from the wizard. See [Chapter 5, "Oracle Reports Services Security with Oracle Portal"](#) 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 Services Launcher is an example of an ActiveX container. Refer to the ActiveX and Oracle Reports Services Launcher online help for more information.
- The SRW.RUN_REPORT is a packaged PL/SQL procedure that runs an Oracle Reports Services 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 Reports Services 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 the Oracle Reports Services 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 Services Queue Manager), then the runtime engine will run the other report normally.

Refer to [Appendix B, "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 Services 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 Services Queue Manager.

Refer to the Oracle Reports Services Queue Manager online help, or see [Appendix C, "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 an Oracle Portal component. See [Chapter 5, "Oracle Reports Services Security with Oracle Portal"](#) 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` (CGI) environment variable on the Web server machine specifies the name of a valid key map file. See [Appendix D, "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` (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 B, "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 D, "Environment Variables"](#) for more information.

3. Add or update the hyperlinks on your Web page. See [Section 4.4.2, "Specifying a Report Request from a Web Browser"](#).

4.4 Specifying Report Requests

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 C, "Oracle Reports Services Configuration Parameters"](#).

The reports source path can also be set in the `REPORTS60_PATH` environment variable. See [Appendix D, "Environment Variables"](#) for more information.

Start the Oracle Reports Services 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 Reports Services Builder online help for more information about building a report. To access Oracle Reports Services Builder only help, click on the  icon and do the following steps:



1. For online help on this task, choose **Help** → **Report Builder Help Topics**.
2. On the Index page, type the following:
report, building
3. Then click **Display** to view the following 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 an Oracle Portal component. See [Chapter 5, "Oracle Reports Services Security with Oracle Portal"](#) 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 B, "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 D, "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 G, "Troubleshooting"](#) for more information.

4.4.3 Scheduling Reports to Run Automatically

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

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

If you publish your reports on an Oracle Portal site as an Oracle Portal 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, "Oracle Reports Services Security with Oracle Portal"](#) for more information.

Oracle Reports Services Security with Oracle Portal

Oracle Reports Services uses Oracle Portal to perform a security check that ensures that users have the necessary privileges (access control) to run reports on restricted Oracle Reports Services servers and printers. Access control determines the following:

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

Oracle Portal stores information about the RDF (that is, how to run the report) as a packaged procedure. In order to run a report, Oracle Portal also needs to store access control information about the restricted Oracle Reports Services server 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 Oracle Portal.

You can make report requests available to users on the Web by doing the following:

- Adding a link as an Oracle Portal component to an Oracle Portal content area that points to the report's packaged procedure.
- Scheduling a request to run automatically and push the report output to an Oracle Portal content area for users to view.
- Adding standard URL syntax to a Web content area as a hyperlink.

5.1 Overview

There are two levels of security that need to be managed:

- Database-level security
- Application-level security

Database-level security defines the users or roles that can access data within the database. The DBA grants this security. This database-level security must be established and in place when configuring your reports environment.

To further define your reporting environment, application-level security can be put in place to specify which report requests the users or groups can generate. This application-level security is very important to ensure only those authorized users or groups can generate a specific report.

Oracle Reports Services *6i* introduces an open infrastructure for report administration and security with an out-of-the-box implementation using Oracle Portal release 3.0 (previously known as Oracle WebDB). All Oracle Portal content is stored in a repository in an Oracle database, making the access control data easy to enter, backup, and retrieve. This chapter discusses configuring and establishing your security policies for deploying Oracle Reports Services *6i* via Oracle Portal release 3.0. All Oracle Portal users are lightweight users and have application-level security privileges. They do not have database-level security privileges.

With the Oracle Reports Services out-of-the-box implementation, Oracle Portal stores the application-level access control data that can be utilized by any existing Oracle Reports Services server. The deployment model is open to allow for access to report generation via an Oracle Portal content area or a custom portal. In either scenario, the security access control data stored within the Oracle Portal repository is used for authorizing an end user. Both Oracle Reports Services and Oracle Portal are part of your license for Oracle*9i* Application Server.

When deploying your Oracle Reports Services outside of an Oracle Portal content area, you might utilize the security access control data stored in Oracle Portal by passing report requests through a registered, secured Oracle Reports Services server. You can do this by using either of the following:

- RWCGI60 (an Oracle Reports Services CGI executable passed through a URL) or a servlet
- RWCLI60 (an Oracle Reports Services command line interface to an Oracle Reports Services server)

If you choose another security model other than Oracle Portal, an open C API is provided so you can write a custom link to your own access control server. The API can be rewritten to talk to another security server. (for example, to look at a custom security schema written in an Oracle database). Doing this requires that you write a C interface as detailed in the following steps. The instructions assume that your security implementation has already been created; if it has not, then this must be done first. There are separate instructions for Windows and UNIX environments.

5.1.1 Creating a Security DLL for Oracle Reports Services 6i Security in a Windows Environment

The following steps are necessary to create a security DLL for Oracle Reports Services 6i security in a Windows environment:

1. Using a Win32 C compiler (for example, Visual C++), create a new project and specify that you will be creating a DLL.
2. Create a file called RWKSS.C and include the RWKSS.H header file, which is located in the %ORACLE_HOME%\report60\server\security directory.
3. Implement the functions named in the RWKSS.H header file, which are called by the Oracle Reports Services server, to perform the security check against your security repository.
4. Create the DLL and link in the RWK60.DEF file, which contains the list of exported functions. This file is located in the %ORACLE_HOME%\report60\server\security directory. Give the DLL the file name 'rwk60.dll'.
5. Rename the existing RWK60.DLL, which is located in the %ORACLE_HOME%\bin directory, and copy your DLL to this location.
6. Restart the Oracle Reports Services server.

5.1.2 Creating a Security Library for Oracle Reports Services 6i Security in a UNIX Environment

The following steps are necessary to create a security library for Oracle Reports Services 6i security in a UNIX environment:

1. Create a file called `rwkss.c` and include the `rwkss.h` header file, which is located in the `$ORACLE_HOME/reports60/pub` directory.
2. Implement the functions named in the `RWKSS.H` header file, which are called by the Oracle Reports Services server, to perform the security check against your security repository.
3. Compile and make a dynamic library. For example:

```
cc -c rwkss.c
ld -dy -G -o rwk60.so rwkss.o
```

4. Check that the library is dynamic by entering the following:

```
file rwk60.so
```

Ensure the response says that the library is dynamic.

5. Rename the existing `rwk60.so`, which is located in the `$ORACLE_HOME/bin` directory, and then put your new dynamic library in this location.
6. Restart the Oracle Reports Services server.

5.2 Database-Level Security

Database-level security is what determines if you have access to the data within a specified database. You can store a user name and password in the key mapping file (`cgicmd.dat`) or you can be prompted for the specific user ID and password.

Unless the user name and password are hard coded into the key map file (or supplied as part of the URL), any user accessing Oracle Reports Services is required to identify themselves for authentication purposes. As the HTTP release 1.0 protocol is stateless (that is, each call to the server is effectively independent of all others), it would result in the user needing to authenticate themselves for each report request.

To solve this issue and to allow you to authenticate only once, the report makes use of client-side cookies to store the required authentication information within the browser for the current session. Once you are authenticated, an encrypted cookie is created in the browser, allowing for multiple report jobs to be submitted without the need to re-authenticate at each request.

Note: If there is a requirement to force a re-authentication on the submission of a given report, use the `SHOWAUTH` and `AUTHTYPE` command line arguments or include a `%D` in the respective report entry in the key map file (use of `%D` forces the you to reenter you user name and password each time the report is called).

Within a given Web application, you frequently access reports that run against multiple instances of an Oracle database (or ODBC data sources). To minimize the number of times a you must be authenticated (once to each different server), an encrypted cookie is created. The cookie contains database authentication information for many database instances, allowing connections to multiple instanced of an Oracle database.

Database connection information is supported by specifying the `USERID` parameter. For example, when a report is submitted using `USERID=<$username>` in conjunction with a Net8 database alias, then connections are created in the browser for each referenced database instance. For example, the following key map file entries would result in you being authenticated against two different database instances through one encrypted cookie created in the browser (this cookie is for both `ORCL` and for `PROD`):

```
Rep1: report=Rep1.rdf userid=$username@ORCL destype=CACHE desformat=HTML
```

```
Rep2: report=Rep1.ref userid=$username@PROD destype=CACHE desformat=HTML
```

With any subsequent request, the user name and password are retrieved from the appropriate cookie and used to authenticate you against that database. If no connect string is defined in the command line (that is, a user ID is not specified in the command line), then the Oracle Reports Services CGI executable uses the last database connect string that achieved a successful connection.

The cookie is removed when you close the browser session, but it might also be important to limit the lifetime of the cookie within a given session. For example, you might have logged in and then gone to lunch, leaving the browser session open for an extended period of time. To control this type of security breach, the administrator can define the `REPORTS60_COOKIE_EXPIRE` environment variable for the CGI or servlet. When the Oracle Reports Services executable receives a job request from the client, it compares the time saved in the cookie with the current system time. If the time is longer than the number of minutes defined in the environment variable (for example, 30 minutes), then the cookie is rejected and you are again required to identify yourself for authentication. The following table shows the environment variables that affect database user authentication:

Table 5–1 Environment Variables for User Authentication

<code>REPORTS60_COOKIE_EXPIRE</code>	<p>Determines the expiration time of the cookie, in minutes, for the Oracle Reports Services CGI or servlet. The default value is 30.</p> <p>Cookies save encrypted user names and passwords on the client side when you log into a secured Oracle Reports Services server to run report requests. When you successfully log in, the browser is sent an encrypted cookie. When a cookie expires, you must be re-authenticated to run subsequent report requests (that is, ones that are sent to a secured Oracle Reports Services server).</p>
--------------------------------------	--

REPORTS60_DB_AUTH	<p>Points to an HTML file that sets the database authentication window template name used to log into the database, but no the entire path since it is placed in the following directory for Windows NT:</p> <p><code>%ORACLE_HOME%\REPORT60</code></p> <p>For UNIX it is placed in the following directory:</p> <p><code>\$ORACLE_HOME/reports60</code></p> <p>The default value is <code>dbauth.htm</code>.</p> <p>Using the <code>REPORTS60_DB_AUTH</code> environment variable allows you to customize the database authentication HTML form.</p>
REPORTS60_ENCRYPTION_KEY	<p>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 <code>reports6.0</code>.</p>

5.3 Application-Level Security

Application-level security is a requirement to ensure that you have the appropriate access to the resources needed to run particular reports. This does not mean that you have access to the data in the database (that is, database-level security privileges). The authorization scheme for application-level security necessitates the following criteria:

- Who has access to each report.
- When can a report be run.
- Which servers or printers can be accessed to run or print the report and when it can be accessed.
- Which parameters a particular user can use with a particular report.

5.4 Integration with Oracle Portal

Oracle Portal is a browser-based, Web content publishing and developing solution that allows end users and developers to instantly publish information and build data-driven departmental portals.

Oracle Portal release 3.0 is tightly integrated with Oracle Reports Services to create a robust and secure reporting environment. New wizards have been added to Oracle Portal for Oracle Reports Services security, permitting an authorized user to define access controls to reports, Oracle Reports Services servers, printers, output formats, and report parameters.

The content area building capabilities of Oracle Portal provide an easy mechanism with which to publish reports for end user access via the Web, though this is not a requirement for publishing your reports via the Web.

Once the access control information is defined within Oracle Portal, it is stored in the Oracle Portal repository. As an Oracle Portal user, you can then, optionally, add the registered RDF to be accessed from an Oracle Portal content area. As an Oracle Portal user, you can make a request to run a given report, the Oracle Portal repository is used to verify the Oracle Portal your access privileges to run a particular report using the specified Oracle Reports Services server. If you are not utilizing Oracle Portal to publish your reports, you can still take advantage of the security model to secure all of your reports. You can easily accomplish this by following the steps outlined in [Section 5.16, "Publishing Report Outside of Oracle Portal"](#).

Oracle Reports Services leverages the Oracle Login Server Single-Signon feature and the concept of lightweight users. Each Oracle Portal page can include data from any different portlet providers, each of which can have their own login procedures. To prevent you from being constantly confronted with user ID requests for each portlet provider, Oracle Portal provides a single-signon feature. When you log in, Oracle Portal automatically logs you into all registered portlet providers and subsystems. Refer to the Oracle Portal documentation for more information about Login Server Single-Signon.

In Oracle Portal release 2.2, users were synonymous with the database user accounts. By default, an Oracle Portal release 2.2 developer could create a component or object in his own database schema. In Oracle Portal release 3.0, users are typically mapped to a database schema for administrative purposes only. The ability to create a component in Oracle Portal release 3.0 no longer depends on whether the developer has privileges to build components in a schema, but instead on whether the developer has privileges to build a component in an application.

Groups replace roles in Oracle Portal release 3.0. A group is a collection of users or other groups that share a common interest or responsibility, and, therefore, have common privileges. Anyone who is logged into Oracle Portal can create a group, not just the Oracle Portal administrator.

How the Oracle Portal users or groups are defined within Oracle Portal defines the accessibility of a particular function or object either from within an Oracle Portal content area or your own custom portal. Since Oracle Reports Services security borrows Oracle Portal users and groups to implement authentication and authorization (meaning Oracle Reports Services security defines who can access what), Oracle Reports Services security can still answer Access Control List (ACL) check questions if you are using your own custom portal.

Refer to the Oracle Portal release 3.0 documentation for more information about users and groups.

5.4.1 Sharing Authentication Information Between Oracle Portal and Oracle Reports Services Servers

Before configuring your security environment, you need to be familiar with the AUTHID command line argument. You use the AUTHID command line argument to authenticate an application user. The AUTHID command line argument is not an Oracle Portal-specific parameter. If you want to run a report against a secure Oracle Reports Services server, then this authentication information is required.

Oracle Portal integration uses the information that was entered when you logged into Oracle Portal. The Oracle Reports Services CGI uses the Oracle Portal user name and session ID as a replacement for the AUTHID command line argument when running the report from within Oracle Portal or outside of Oracle Portal. This works for both Oracle Portal and the content area builder.

5.5 Oracle Portal Integration Architecture

The Oracle Reports Services Web configuration and components remain the same as in previous releases, with the ability to execute reports through the CGI or servlet interfaces. The communication between Oracle Reports Services and the Oracle Portal repository is accomplished via a C API, which by default communicates with the Oracle database, where the Oracle Portal repository resides. The Oracle Portal repository is examined to validate Oracle Portal users and to check for accessibility of the report requests.

Because this architecture employs the use of an open API, you can choose to re-implement the security checks against your own security system. This openness permits you to authenticate users against your Lightweight Directory Access Protocol (LDAP) server or any other custom security server set in place. Refer to the Oracle Portal documentation for more information.

5.6 Installing Oracle Reports Services Security in Oracle Portal

This section describes how to install Oracle Reports Services security and Oracle Portal on one machine. Oracle Reports Services security and Oracle Portal can also be installed on separate machines. They do not have to reside on the same machine to take advantage of the functional security model in place via Oracle Portal. Refer to [Chapter 6, "Configuring Oracle Reports Services Server Clusters"](#) for information about configuring Oracle Reports Services servers and Oracle Portal on multiple servers.

Following are the steps to install Oracle Reports Services *6i* security and Oracle Portal.

- Step 1: Install Oracle Portal into an Oracle database.
- Step 2: Install Oracle Reports Services.

Refer to the *Oracle9i Application Server Installation Guide* for more information about installation.

5.6.1 Step 1: Installing Oracle Portal Into an Oracle Database

Install Oracle Portal release 3.0 into a separate `ORACLE_HOME` with an Oracle database release 8.1.6 or higher. Oracle Portal is an option to the Oracle database that can only be accessed via a Web browser. Oracle Portal release 3.0 is installed through the Oracle9i Application Server. Refer to the *Oracle9i Application Server Installation Guide* for more information.

5.6.2 Step 2: Installing Oracle Reports Services

Install the Oracle Reports Services component. The installer automatically analyzes the dependencies for your machine and then configures the Oracle Reports Services based on the options you choose. Oracle Reports Services is installed through the Oracle9i Application Server.

If you need to use the Oracle Reports Services Builder, then you need to installed it through the Oracle9i Developer Suite.

5.7 Configuring the Security Environment

To configure your security environment for Oracle Reports Services 6i, perform the following steps:

- ❑ Step 1: Enable Oracle Reports Services security within Oracle Portal.
- ❑ Step 2: Add SECURITYTNSNAMES and PORTALUSERID parameters.
- ❑ Step 3: Start Oracle Portal.

5.7.1 Step 1: Enabling Oracle Reports Services Security within Oracle Portal

You can manually enable Oracle Reports Services security within Oracle Portal by running the `RWWWVINS.SQL` script found in the following directory for Windows NT:

```
%ORACLE_HOME%\REPORT60\SERVER\SECURITY\3.0
```

You can manually enable Oracle Reports Services security within Oracle Portal by running the `RWWWVINS.SQL` script found in the following directory for UNIX:

```
$ORACLE_HOME/reports60/server/security
```

Run the following script as the Oracle Portal administrator:

```
sqlplus> @rwwwvins.sql <portal30 schema owner>
```

When installing, you are prompted to enter the following for your Oracle Portal schema owner (for example, `portal30/portal30@orcl`):

```
username/password@connectstring
```

This script creates the appropriate object definitions, menu entries, and groups. The following groups are created:

- RW_ADMINISTRATOR
- RW_DEVELOPER
- RW_POWER_USER
- RW_BASIC_USER

These four groups are created when enabling Oracle Portal and Oracle Reports Services security. Each Oracle Portal user, for which the security authentication is checked, must be assigned to one of these groups.

5.7.1.1 RW_ADMINISTRATOR

An Oracle Portal user assigned to this group (for example, an Oracle Reports Services administrator, an Oracle Portal administrator, or a Login Server administrator) can CREATE, UPDATE, and DELETE the registered report definition files, servers, and printer objects in Oracle Portal. The Oracle Reports Services administrator can assign security privileges for other people and receive full error messages from Oracle Reports Services. Refer to the Oracle Portal documentation for information about how to create and manage a user.

This user also has access to the administrator's functionality in Oracle Reports Services Queue Manager, which means they can manage the server queue, including rescheduling, deleting, reordering jobs in the server, and shutting down a server.

5.7.1.2 RW_DEVELOPER

In addition to the privileges of the RW_POWER_USER and RW_BASIC_USER groups, an RW_DEVELOPER can run all of the CGI commands, such as SHOWENV and SHOWMAP, which show the system environment. This group might be assigned to a developer who needs to do testing and needs to retrieve detailed error messages.

5.7.1.3 RW_POWER_USER

In addition to the privileges of the RW_BASIC_USER group, an Oracle Portal user with RW_POWER_USER group privileges can see more detailed error messages if the security check fails. For example, the message received if they try to run to HTML and this is not permitted might be:

```
Cannot run report to HTML
```

5.7.1.4 RW_BASIC_USER

When Oracle Portal creates an Oracle Portal user, they automatically become part of the RW_BASIC_USER group. An Oracle Portal with these privileges can only run a report if they have been given the privilege to run it. This Oracle Portal user can see very simple error messages should the security check fail. The message received is:

```
Security Check Error
```

Note: With Oracle Reports Services 6i and Oracle Portal 3.0, the Authentication Cookie Domain is no longer used. Oracle Reports Services uses a PL/SQL call to pass the session ID and the user ID to the CGI or servlet.

5.7.2 Step 2: Adding SECURITYTNSNAMES and PORTALUSERID Parameters

This step is done by a user with database-level security privileges.

You must first shut down the Oracle Reports Services server if it is running. Then add the `SECURITYTNSNAMES=<"tnsname">` parameter and the `PORTALUSERID=<portal_username>/<portal_password>` in the Oracle Reports Services server configuration file (for example, `rep60_<machinename>.ora`) found in the following directory for Windows NT:

```
%ORACLE_HOME%\REPORT60\SERVER
```

For UNIX, this configuration file is found in the following directory:

```
$ORACLE_HOME/reports60/server
```

where:

<code>tnsname</code>	is the TNSname of the instance where Oracle Portal is installed.
<code>portal_username</code>	is the name of the database user where Oracle Portal is installed.
<code>portal_password</code>	is the password of the database user where Oracle Portal is installed.

Ensure you have the correct alias in the `tnsnames.ora` file on the machine where Oracle Reports Services is located.

Oracle Reports Services server requires the `SECURITYTNSNAMES` and `PORTALUSERID` entries to know where to look for the access control information when a user submits a job request. The server must be told the name of the database instance in which Oracle Portal and the Oracle Reports Services security framework are installed. Once the `SECURITYTNSNAMES` and `PORTALUSERID` entries have been added to the Oracle Reports Services server configuration file, the access control information in the Oracle Portal repository is enforced. Oracle Portal users who request to run a report against this Oracle Reports Services server are now required to identify themselves.

As an Oracle Portal, when you successfully log into an Oracle Portal content area to run your reports, this login information (user name and session ID) is used as the alternative to the AUTHID command line parameter and verified by the Oracle Reports Services server via Oracle Portal.

5.7.3 Step 3: Starting Oracle Portal

Start Oracle Portal through your Web browser and log into Oracle Portal as the user you identified during the installation. This user has application-level security privileges.

5.8 Printer Access

In your environment, you can have many networked or local printers accessible to your Oracle Portal users. However, you might want to confine Oracle Portal users to a subset of those printers, constraining the use of the printer for certain periods of time, or identify a particular printer to be used for printing output of certain reports. For example, you can have a monthly report that is very lengthy and for which you want output generated to only a fast, high-volume printer in the information technology (IT) department.

Printer access stores information about the following:

- What printers are available to print report output from within Oracle Portal.
- Who has access privileges to print report output.
- When the printer is available for processing by assigning an availability calendar.

As with availability calendars, it is not a requirement to register a printer within the security framework of Oracle Portal.

Once printers are registered within Oracle Portal, you can choose to associate them with an Oracle Reports Services server. Many printers can be registered. However, only printers associated with a particular Oracle Reports Services server are available to print when you register an RDF file and choose to print to a printer.

When defining access to an RDF, you can choose to restrict even further the registered subset of printers to which the report output can be sent. For example, an Oracle Reports Services server might be connected to the printer in the office of the CEO, but it should not be a selection by employees running the general ledger report unless it is the CEO who is running the report. This subset of printers can then be listed to the Oracle Portal user running a report request to select where output should be sent.

5.9 Creation of an Oracle Portal Content Area

Note: All of the information that follows is for Oracle Portal users. All Oracle Portal users have application-level security privileges, but not database-level security privileges.

Oracle Portal provides a creation wizard to step you through the automatic creation of a Web content area, which is contained entirely within the Oracle Portal repository. To create a content page, log into Oracle Portal and click on the Oracle Portal **Navigator** icon, click on the **Content Area** tab, and click on the **Create** button. When creating the content area, a content owner or DBA can add items or links to the Oracle Portal content area. An item could be a URL, a text item, an image map, a file, a PL/SQL call, a link to a folder, or any other Oracle Portal component. Once registered within Oracle Portal, an RDF is treated as any other Oracle Portal component and can be added in the same way to your Web content area.

You can choose to have this link run the report immediately, where the user is authenticated via Oracle Portal and output is generated in the authorized or chosen format. Alternatively, you can choose to schedule the report and push the output to an existing Oracle Portal content area. Refer to [Appendix A, "Controlling User Access to Reports by Defining Calendars"](#) for more information about scheduling reports.

5.10 Setting Up and Deploying a Report

Once you have installed Oracle Reports Services security and Oracle Portal, and set up an Oracle Portal content area, you can begin setting up a user and deploying a report through Oracle Portal. The following sections take you through the steps necessary to set up an Oracle Portal user and how to deploy a report:

- Creating and enabling an Oracle Portal user to administer security.
- Setting up access controls in Oracle Portal.
- Registering a report.
- Deploying a report.
- Running a report.

The following assumptions are made:

- An Oracle Reports Services server has already been installed and configured for Web reporting and can be reached through a URL. For example:

```
http://mycomputer.domain/cgi-bin/rwcgi60.exe
```

- An Oracle Portal content area already exists. This content area is accessed with the Oracle Portal Navigator.
- The `SECURITYTNSNAME=<"tnsname">` parameter has been added to the Oracle Reports Services server configuration file. The `tnsname` references the instance where Oracle Portal is installed.
- The `PORTALUSERID=<portal_username>/<portal_password>` parameter has been added to the Oracle Reports Services server configuration file, where `<portal_username>/<portal_password>` is a valid user name and password of the database where Oracle Portal is installed.

5.11 Creating and Enabling an Oracle Portal User to Administer Security

This step needs to be performed for any Oracle Portal user that can register reports, servers, or printers, and authorize or grant other Oracle Portal users access to these objects.

Overview

This example covers giving an Oracle Portal user the ability to administer Oracle Reports Services security by granting privileges, assigning the DBA group, and assigning the RW_ADMINISTRATOR group.

Assumptions

The following assumptions are made for this example:

- The Oracle Portal administrator opens Oracle Portal using the appropriate URL.
- The Oracle Portal administrator logs in.

5.11.1 Creating and Enabling User REPORTSDEV to Administer Security

The following steps must be performed:

1. From the **Oracle Portal** home page, click on the **Administer** tab.
2. From the **Administer** page, click on **Create New Users** from the **Users** portlet.
The following screen appears:

Create User

Create User ?

User Details

Enter the user name, password, and e-mail address for this user. Confirm the password to make sure you entered it correctly. Passwords are case-sensitive, and the values you enter in the Password and Confirm Password fields must be exactly the same (including case).

User Name

Password

Confirm Password

E-mail Address

3. Under the **User Details** heading, type **REPORTSDEV** in the **User Name**, **Password**, and **Confirm Password** fields. Then click on the **Create** button. The following screen appears:

Create User Home

Click [REPORTSDEV](#) to edit the user.

Create User ?

User Details

Enter the user name, password, and e-mail address for this user. Confirm the password to make sure you entered it correctly. Passwords are case-sensitive, and the values you enter in the Password and Confirm Password fields must be exactly the same (including case).

User Name

Password

Confirm Password

E-mail Address

- Click on **REPORTSDEV** to edit the user. The following screen appears:

Edit User Home

Edit User: REPORTSDEV

Main Preferences Contact Info Privileges

Delete Apply OK Cancel

Personal Details

Enter the user's first, middle, and last names, employee number, and e-mail address.

First Name

Middle Name

Last Name

Employee Number

E-mail Address

- Scroll through this screen to the **Group Membership** heading and select the **DBA** and **PORTAL_ADMINISTRATORS (Non-DBA Privileged Administrators)** groups as shown in the following screen:

Group Membership

Select the groups to which to add this user. The groups determine what tasks the user is able to perform.

- DBA (Database Administrators)
- PORTAL_ADMINISTRATORS (Non-DBA Privileged Administrators)
- PORTAL_DEVELOPERS (This group has the privileges for developing applications)
- PORTLET_PUBLISHERS (This group has the privilege of publishing portlets)

- Click on the **Apply** button.

7. Click on the **Privileges** tab. The following screen appears:

Edit User: REPORTSDEV

Main Preferences Contact Info Privileges

Delete Apply OK Cancel

Object Privileges

Choose the privileges to grant to this user for each object. The privileges that you grant for a particular object apply to ALL the objects of that type. You only see the global privileges that you have the privilege to manage.

Page Privileges

Object Type	Privileges
All Pages	None
All Styles	None
All Layouts	None
All Providers	None

8. Scroll through this screen to the **Content Areas Privileges** heading and select **Manage** from the **Privileges** drop down list.
9. Continue scrolling through this screen to the **Application Privileges** heading and select **Manage** from the **Privileges** drop down list for **All Applications** and **All Shared Components**.
10. Click on the **Apply** button.
11. Click on the **OK** button. You are returned to the **Administer** page.
12. Click on the **List** button to select the **DBA** and **RW_ADMINISTRATOR** groups from the list.

Note: You can select more than one object by holding the **Ctrl** key down and clicking on your choices.

13. Click on the **Edit** button. The following screen appears:

Edit Group: RW_ADMINISTRATOR

Edit Group Details
Change the group's name, or supply a description of the group.

Name

Applies To

Description

14. Click on the **Members** tab. The following screen appears:

ADMINISTRATOR

Group Members
to add a user to this group, or click to add an another group as a member. You can also type in the name of the user or group directly To Members List to add the user or group to the list below.

Member

Name

15. Under the **Group Members** heading, type **REPORTSDEV** in the **Name** field, or click on the **Browse Users** button to select REPORTSDEV from the list.

16. Click on the **Add Members to List** button. After doing this, scroll through this screen to the **Group Member List** heading and you can see that REPORTSDEV has been added. You have now given REPORTDEV privileges for the DBA and RW_ADMINISTRATOR groups.
17. Click on the **OK** button. You are returned to the Oracle Portal **Administer** page.
18. You now need to log out as the Oracle Portal administrator so that you can log in as REPORTSDEV and administer security.

REPORTSDEV has now been created and can administer security for Oracle Reports.

5.12 Setting Up Access Controls in Oracle Portal

The integration of Oracle Reports Services and Oracle Portal provides an out-of-the-box administrative interface to restrict access to reports that are run through Oracle Reports Services. The security checks performed ensure that Oracle Portal users have the necessary access control.

Keep in mind that the access control data stored in the Oracle Portal repository refers to the functional or application-level security, that is the ability of an Oracle Portal user to access a particular report. The data security can be handled through the USERID parameter, can be passed at runtime, or the Oracle Portal user can be prompted.

All of the utilities employ wizards for creating, editing, or deleting access control information. Once entered, the Oracle Portal repository stores the access control information as metadata. Only those Oracle Portal users who have Oracle Reports Services system administrator privileges (the DBA and RW_ADMINISTRATOR group) can access this security information in Oracle Portal.

5.13 Registering a Report

Now that REPORTSDEV has been created with the ability to administer security, he can do the following:

- Secure reports
- Secure servers
- Secure printers
- Define availability calendars
- Authorize users to run and access reports, servers, and printers

Overview

This example walks you through the following:

- Registering a server
- Creating RDF access

Assumptions

The following assumptions have been made for this example:

- Oracle Portal has been opened using the appropriate URL for your Oracle Portal installation.
- REPORTSDEV is logged into Oracle Portal.
- REPORTSDEV is registering a report called **accounting.rdf**.
- REPORTSDEV is authorizing users SCOTT and BJ (who are already Oracle Portal users) to run the report.
- An Oracle Reports Services server called PUBSVR has already been set up and is available.

5.13.1 Registering a Server

Oracle Reports Services server access stores information about the following:

- What Oracle Reports Services servers are available for processing job requests.
- What registered printers are available for printing output through the registered Oracle Reports Services server.
- Who has privileges to submit report requests to a given Oracle Reports Services server.
- When the Oracle Reports Services server is available for processing job requests.
- Whether a given Oracle Reports Services server can run any report or only those reports that have been registered for secure access.

The actual Oracle Reports Services server within the Oracle Portal framework already exists and must be configured to run report requests.

Do the following to define server access:

1. From the Oracle Portal home page, click on the **Administer** tab.
2. Click on **Oracle Reports Security Settings** from the **Oracle Reports Security** portlet.
3. Click on **Create Reports Server Access**. The following screen appears:

Step 1 of 3

Next > Cancel

Server Name and Printers

The Server Name is used to identify the server within Oracle Portal.

Server Name

Reports Server TNS Name

Description

4. In the **Server Name** field type **NEWSERVER**.
5. In the **Reports Server TNS Name** field type **PUBSVR** (this name is the name of your Oracle Reports Services server).
6. In the **Description** field type **Local reports server**.
7. In the **Oracle Reports Web Gateway URL** enter the location of the Oracle Reports Services CGI or servlet. For example:

`http://mycompany.docmain/cgi-bin/rwcgi60.exe`

8. Leave **Run Only Registered Report Definition Files** and **Printers** blank. They are not being created for this example. You are not choosing any printers for this example. However, if you had registered printers, then you could associate one or more printers with this Oracle Reports Services server.

Note: If you put a check mark in the **Run Only Registered Report Definition Files**, then you are telling the Oracle Reports Services server not to run any reports that have been secured within Oracle Portal.

9. If you had secured printers within Oracle Portal, then you could associate one or more printers for this RDF file; however, in this example you have not registered printers and this RDF can be printed on any printer. The **Printers** field can contain printers that were created earlier; however, they are not needed for this example. Refer to [Section 5.8, "Printer Access"](#) for more information about printers.
10. After filling in all the fields, click on the **Next** button. The following screen appears:

Add Server Access Navigator

NEWSERVER

Step 2 of 3 >>>

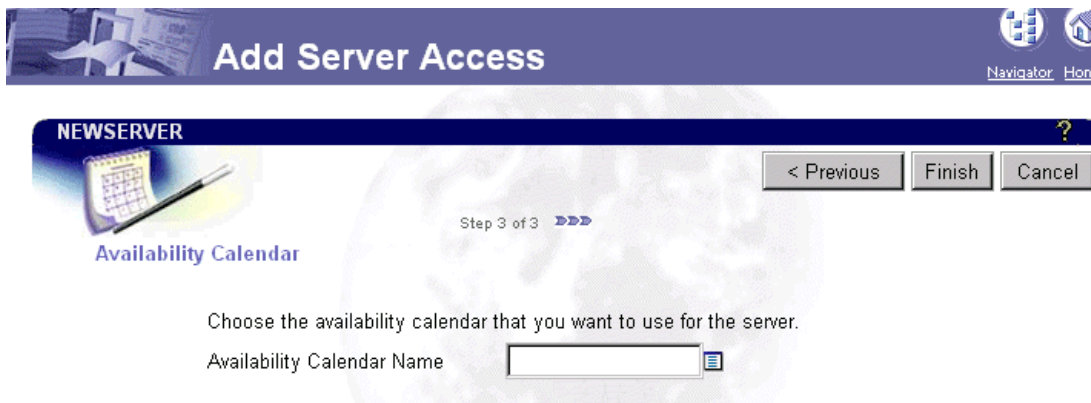
Users and Groups
Select the users and groups that you want to have access to the server.

Grant Access
Enter the name of the user or group to which you want to grant privileges, or click on to select a user, or click on to select a group, then choose the level of privileges you want to grant, then click Add.

Grantee Manage

11. Under the **Grant Access** heading, type **SCOTT (USER)** in the **Grantee** field, or click on the **Browse Users** button to select SCOTT from the list.

12. Select **View** from the pull down list and then click the **Add** button.
13. Scroll to the **Change Access** heading and you see that the user has been granted access to the server.
14. Now you need to grant access to BJ. Repeat steps 11 through 13, typing **BJ (USER)** in the **Grantee** field, or by selecting him from the **Browse Users** list.
15. Click on the **Next** button to continue. The following screen appears:



Leave this screen blank for this example. Refer to [Appendix A, "Controlling User Access to Reports by Defining Calendars"](#) for more information about availability calendars.

Note: If you want to restrict the days or times this server is available for Oracle Portal reporting, then you would create an availability calendar and then select it from here.

16. Click on the **Finish** button. The following screen appears:



17. Click on the **Close** button. You are returned to the **Oracle Reports Security Setting** page.

You have now registered an Oracle Reports Services server. Now you need to register a report.

5.13.2 Creating Report Definition File Access

Report Definition File (RDF) access stores information about the following:

- What Oracle Reports Services files (RDF, REP, REX, and XML) have been registered for secure access. You can put the full path to the report. However, Oracle Corporation recommends that you use just the RDF, REP, REX, or XML without the path as the file name and either add the location to the file in the REPORTS60_PATH environment variable or the SOURCEDIR path.
- What is the name by which this instance of the report is known within Oracle Portal (only applicable within Oracle Portal).
- Who has access privileges to run this instance of the report.

- When this instance of the report is available to run.
- Which Oracle Reports Services servers can be used to process requests for the particular report.
- How the report is delivered with use of parameters, specified formats (PDF, HTML, or HTMLCSS), and validation triggers.
- Which printers this report can print on.

Once an RDF is registered within Oracle Portal it creates an Oracle Portal component. Oracle Corporation does not recommend registering multiple instances of the same RDF file.

Besides designating the Oracle Portal users that have access to specific reports, you might want to specify how Oracle Portal users are to interact with the reports. In addition to the parameters you might have specified in the report, you can also create a parameter form when registering your report in Oracle Portal.

The Oracle Portal parameter form is used to set security restrictions, such as having only limited report output formats that are valid for a given report and having the parameter form only display with these valid formats. The security information is stored in the Oracle Portal repository.

The parameter form for Oracle Reports Services allows you to add additional restrictions, such as attaching a PL/SQL trigger. Oracle Corporation recommends that you use only one of the parameter forms, the Oracle Portal parameter form or the Oracle Reports Services parameter form, to avoid conflicts.

This optional parameter form can be used to restrict the values to which users have access or for any additional parameters needed to run this report. For example, forcing page streaming for an HTML report, displaying data based on specific values, or a defined LOV. Furthermore, you might want to specify which parameters are exposed to a user during job submission, which allows different users to apply different options to the same report. For example, you might want the user to specify whether the report output is HTML, HTMLCSS, or PDF.

In the **Parameter Entry Form**, you can specify whether a parameter is visible along with the values for selection. To do this, you select the **Customize** link from the **Manage Component** screen. You then check those parameters that you would like to make visible to the end user.

You do the following to create RDF access and register your report:

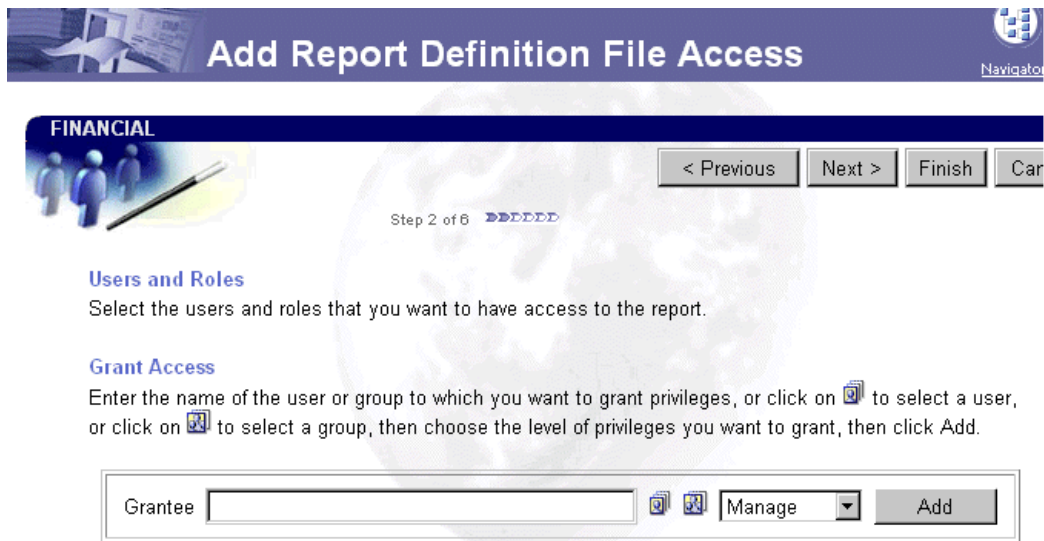
1. Return to the **Oracle Reports Security** screen.
2. Click on **Create Reports Definition File Access** from the **Reports Definition File Access** portlet. The following screen appears:



The **Application**, **Report Name**, and **Reports Server** fields are already filled in for you.

3. Change the **Report Name** field to say **Financial**.
4. Select **NEWSERVER** in the **Reports Server** field. If you have only one Oracle Reports Services server, you must still select it to continue. You can also highlight more than one server by holding down the **Ctrl** button and clicking on each server you want to use.
5. In the **Oracle Reports File Name** field type **accounting.rdf**. It is assumed that the RDF can be found along the **REPORTS60_PATH**; however, you can hard code the full path to it.
6. The **Description** field is optional. In this example, type in **Financial statement**.

7. Click on the **Next** button. The following screen appears:



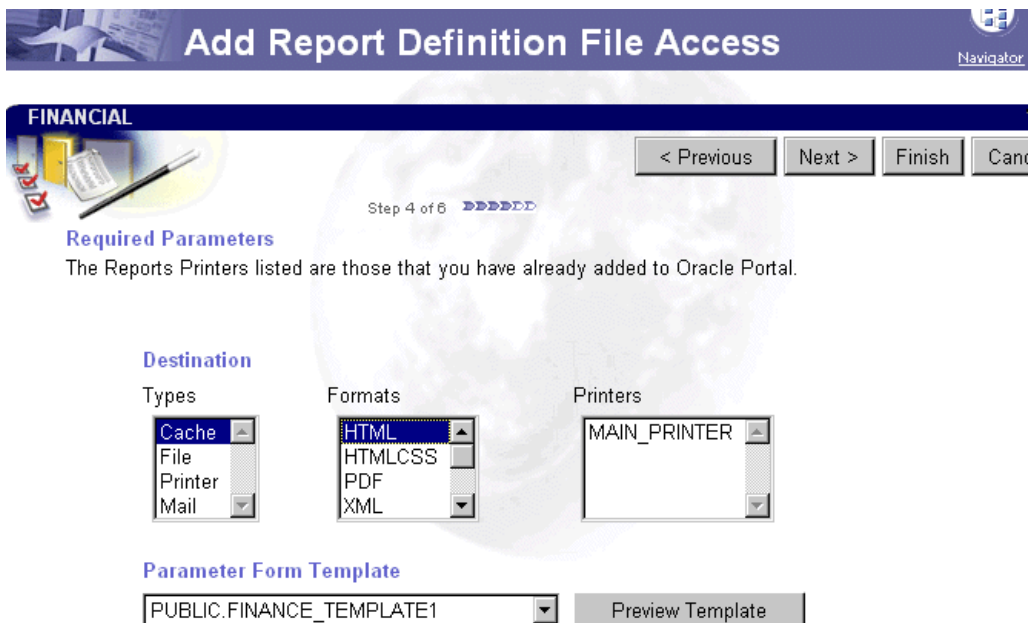
8. Under the **Grant Access** heading type **SCOTT (USER)** in the **Grantee** field, or click on the **Browse Users** button to select SCOTT from the list.
9. Select **Manage** in the drop down list.
10. Click on the **Add** button.
11. Scroll through this screen to the **Change Access** heading and you see that the user has been granted access to the RDF file.
12. Now you need to grant access to BJ. Repeat steps 8 through 11, selecting **BJ (USER)** as the user.

13. Click on the **Next** button. The following screen appears:

The screenshot shows a web interface for configuring report access. At the top, a dark blue banner contains the text "Add Report Definition File Access" and a "Navigator" icon. Below this, a dark blue bar displays "FINANCIAL" and a calendar icon. The main content area is titled "Availability Calendar" and includes the instruction: "Choose the availability calendar that you want to use for the Oracle Reports file." A label "Availability Calendar Name" is positioned to the left of a text input field. To the right of the input field is a small icon of a document with a list. At the top right of the main area, there are four buttons: "< Previous", "Next >", "Finish", and "Cancel". A progress indicator shows "Step 3 of 6" with six dots, the third of which is filled.

Since we do not wish to specify availability, you leave this screen blank.

14. Click on the **Next** button. The following screen appears:



15. This screen lets you select the **Destination** information. These are **Types**, **Formats**, **Printers**, and **Parameter Form Template**. For this example, you select **Cache** for **Types**, and **HTMLCSS** and **PDF** for **Formats**.

Note: You select more than one object by pressing the **Ctrl** key down and clicking on your choices.

16. Click on the **Next** button. The following screen appears:

Add Report Definition File Access Navigator

FINANCIAL

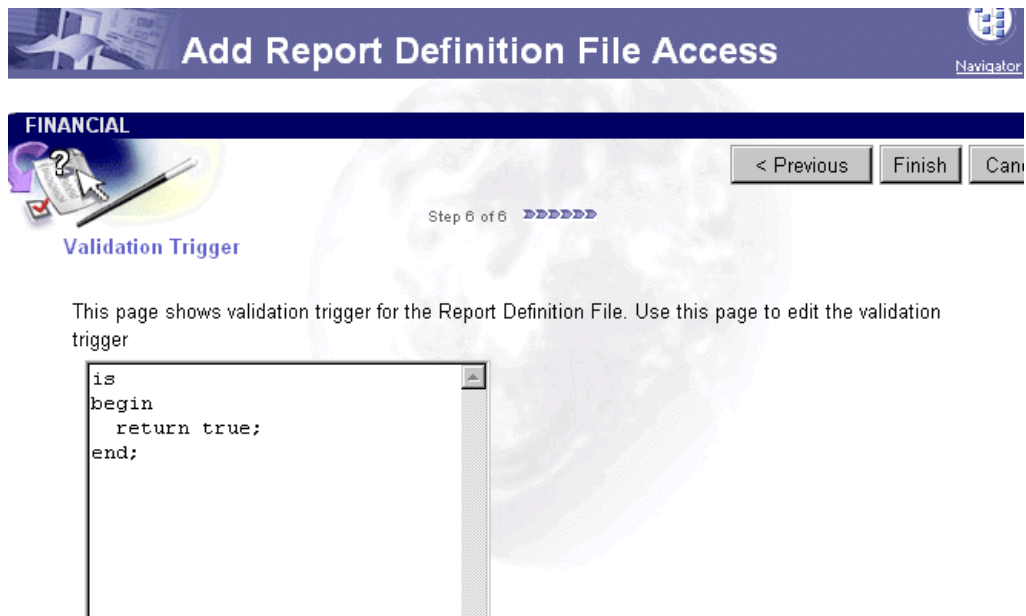
Step 5 of 6 < Previous Next > Finish Cancel

Optional Parameters
 If want to include more parameters, click the 'More Parameters' button and more rows will be added.

Parameter Name	LOV	Low Value	High Value
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

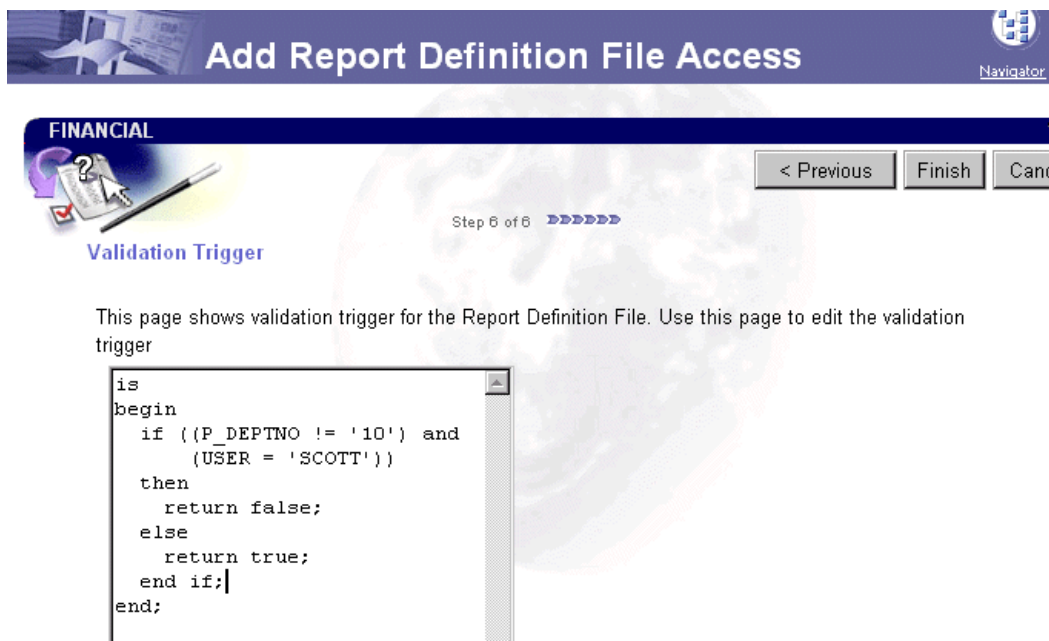
17. For this example, you are going to further restrict access by restricting the department number passed to the report. Type **P_DEPTNO** in the **Parameter Name** field.

18. Click on the **Next** button. The following screen appears:



A validation trigger is used to create conditional restrictions that cannot be defined on either the **Required Parameters** page or the **Optional Parameters** page. Validation triggers are PL/SQL functions. A validation trigger is run when users accept the **Runtime Parameter Form**.

19. You change the information in the **Validation Trigger** screen so that it looks like the following screen. SCOTT can only run this report for department 10. There is no such restriction for other users.



Add Report Definition File Access Navigator

FINANCIAL

Validation Trigger

Step 6 of 6

< Previous Finish Cancel

This page shows validation trigger for the Report Definition File. Use this page to edit the validation trigger

```
is
begin
  if ((P_DEPTNO != '10') and
      (USER = 'SCOTT'))
  then
    return false;
  else
    return true;
  end if;
end;
```

20. Click the **Finish** button. The following screen appears:

Develop: EXAMPLE_APP.FINANCIAL Develop Manage Clos

Report Definition File Access FINANCIAL

Application EXAMPLE_APP

Version(s) Status [1 \(PRODUCTION with VALID Package\)](#)

Last Changed Tuesday 03-OCT-2000 07:56 by REPORTSDEV



Run Link PORTAL30_DEMO.FINANCIAL.show
PORTAL30_DEMO.FINANCIAL.show_parms

PL/SQL source [Package Spec](#), [Package Body](#)

Call Interface [Show](#)

[Edit](#) [Run](#) [Run As Portlet](#) [Customize](#) [Add to Favorites](#) [About](#) [Delete](#)

21. Click on the **Customize** link. The following screen appears:

Oracle Report Parameters  

Visible to user

Server:

Printer:

Destype:

Desformat:

Desname:

P_DEPTNO:

22. On this screen, under the check boxes for **Visible to user**, select **Desformat** and **P_DEPTNO**. In the **P_DEPTNO** field, type **10**.

23. Click on the **Save Parameters** button. You must do this to save the changes you have made. A confirmation appears, '**Parameters Saved**', confirming your changes.

24. Click on the **Run Report** button. The following screen appears:

Database User Authentication

User Name:

Password:

Database:

25. Type **SCOTT** in the **User Name** field, **TIGER** in the **Password** field, and **ORCL** in the **Database** field. Then click on the **Submit** button. The following screen appears, confirming that the report has run successfully.



Report run on: August 2, 2000 11:19 AM

<i>Department No</i>	<i>10</i>	<i>ACCOUNTING</i>	<i>Location</i>	<i>NEW YORK</i>
<i>Empno</i>	<i>Name</i>	<i>Job</i>	<i>Salary</i>	<i>Commission</i>
7782	CLARK	MANAGER	\$2450.00	
7839	KING	PRESIDENT	\$5000.00	
7934	MILLER	CLERK	\$1300.00	

You have successfully registered a report and given privileges to REPORTSDEV (this happens automatically), SCOTT, and BJ.

5.14 Deploying a Report

You are now ready to deploy a report.

Overview

This example walks you through deploying a report in an Oracle Portal content area.

Assumptions

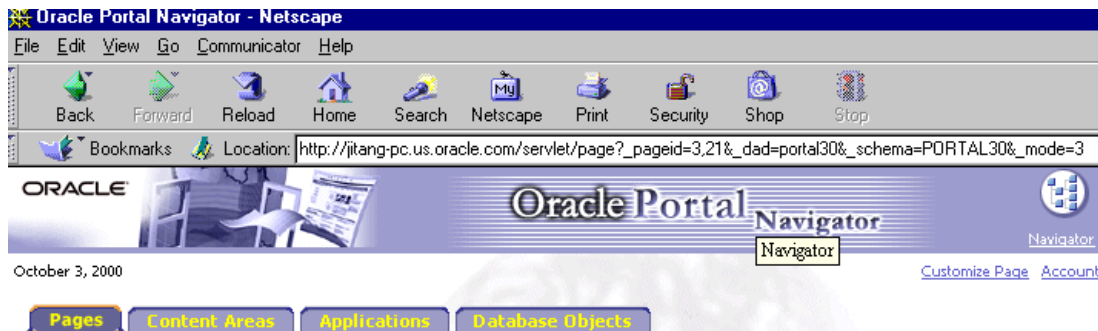
For this example, the following assumptions have been made:

- Oracle Portal has been opened using the appropriate URL for the Oracle Portal installation.
- REPORTSDEV is logged into Oracle Portal.
- A content area called **Reports Security Test** already exists.

5.14.1 Deploying a Report to an Oracle Portal Content Area

Do the following steps to deploy a report to an Oracle Portal content area:

1. From any page, click on the **Navigator** icon. The following screen appears:



Browse the Pages, Page Layouts and Page Styles that are available to you. Top-Level Pages are portal-wide pages controlled by administrator. My Pages are the pages you have created. User Pages are pages created by all users.

Find: Go

- Click on the **Content Areas** tab. The following screen appears:



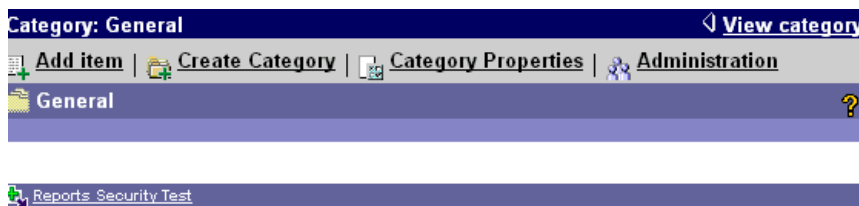
- Click on **Reports Security Test**. The following screen appears:



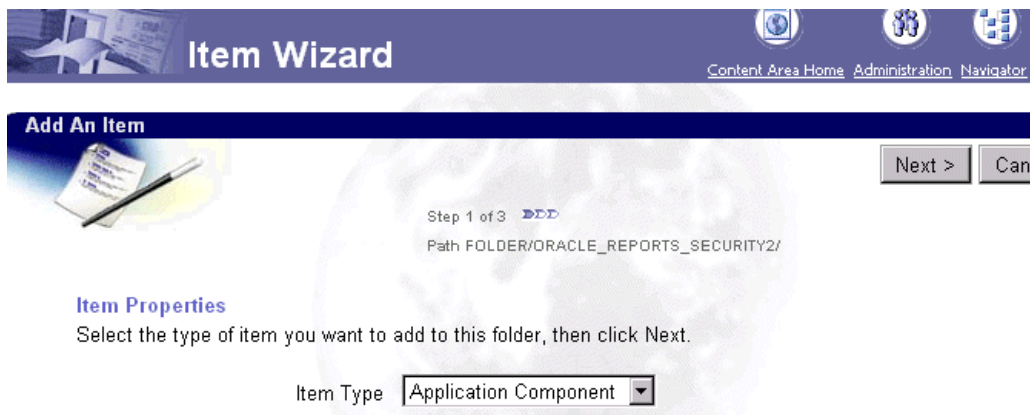
4. Click on **General**. The following screen appears:



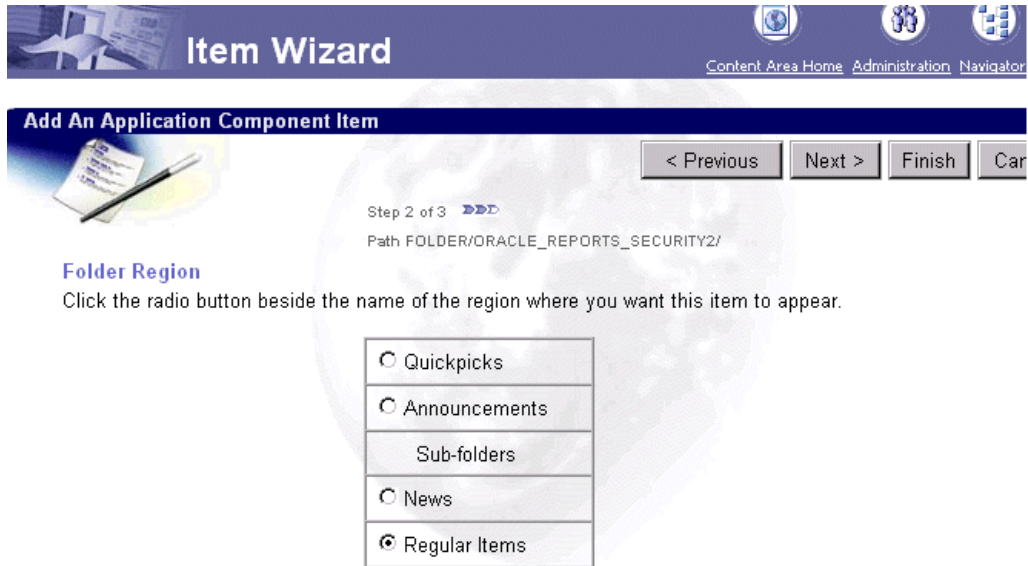
5. Click on **Edit Category**. The following screen appears:



6. Click on **Add item**. The following screen appears:



7. Select **Application Component** from the **Item Type** drop down list. Then click on the **Next** button. The following screen appears:



8. Under the **Folder Region** heading, click the appropriate radio button. In this case, **Regular Items**.
9. Scroll to the **Primary Item Attributes** heading and select **EXAMPLE_APP: FINANCIAL** from the **Application Component** drop down list.
10. Type **Financial report** in the **Display Name** field.
11. Select **Reports Security Test** from the **Folder** drop down list.
12. Select **General** from the **Category** drop down list.
13. Enter **Financial report for a department** in the **Description** field. Then click on the **Next** button. Your screen now looks like the following:

Primary Item Attributes

Select the component to execute when this item is clicked. Enter a display name for the item's link text which appears in the folder area. Select a category that best describes the item's content, then enter a description.

Application Component

Display Name

Folder:

Category

Description

14. Click on the **Next** button. The following screen appears:

Item Wizard
Content Area Home Administration Navigator

Add An Application Component Item

Step 3 of 3 >>>
Path FOLDER/ORACLE_REPORTS_SECURITY2/

Secondary Item Attributes

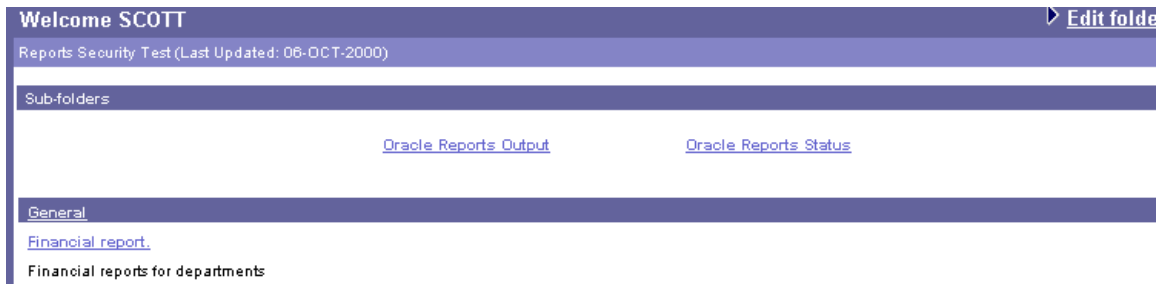
Choose perspective(s) for this item. Enter the location of the image file and select the image alignment. Edit the keyword(s) that will help locate the item during a search, and edit the author name. Set document control on the item so that it may be shared between users via a check-in and check-out process. You can also choose to hide the item in View mode.

Available Perspectives Displayed Perspectives

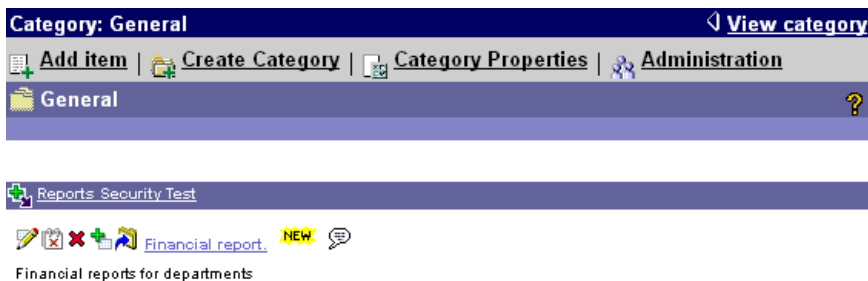
Perspectives

15. There are two ways that SCOTT can display the report.

- The first is to click on the **Financial report** link found by navigating to the **Content Areas**, then to **Reports Security Test** as shown in the following screen:



- The second way gives SCOTT additional options. Continue from Step 14 and scroll to the **Display Options** heading and select **Display Parameter Form**. Then click on the **Finish** button. The following screen appears and you can see that your report has been added.



16. Click on **Financial** report. The following screen appears:

Financial report.

Oracle Reports Parameters/Scheduling

Parameters Schedule

Run Report

Desformat:

P_DEPTNO:

17. Click on the **Run Report** tab. The following screen appears confirming that the report ran successfully.



Report run on: August 2, 2000 11:19 AM

Department No	10	ACCOUNTING	Location	NEW YORK
Empno	Name	Job	Salary	Commission
7782	CLARK	MANAGER	\$2450.00	
7839	KING	PRESIDENT	\$5000.00	
7934	MILLER	CLERK	\$1300.00	

18. REPORTSDEV can now log out.

Your report, FINANCIAL, has now been successfully deployed to the **Reports Security Test** content area in Oracle Portal.

5.15 Running a Report

Now that REPORTSDEV has registered a report, given SCOTT and BJ permission to run the report, and deployed the report to the Reports Security Test content area, the report is ready to be run by the Oracle Portal users.

Overview

This example walks you through the following:

- Finding the report
- Running the report

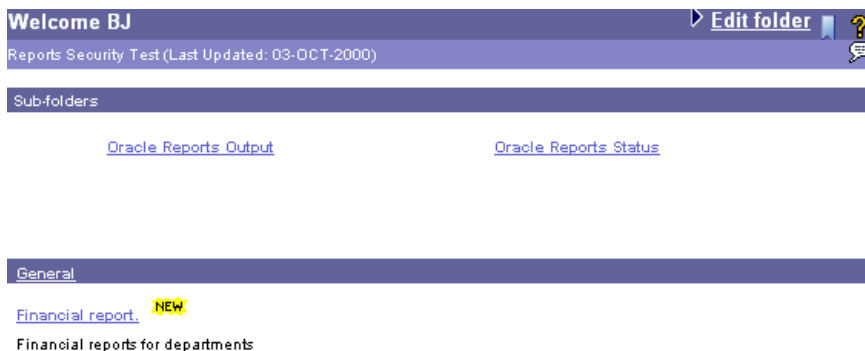
Assumptions

For this example, the following assumptions have been made:

- BJ is logged in.
- BJ has navigated to the Report Security Test content area.
- BJ is going to run a report called **Financial**.

To run the report do the following steps:

1. From the **Content Areas** tab, select **Reports Security Test**. The following screen appears:



- Click on **Financial report**. The following screen appears:

Financial report.

Oracle Reports Parameters/Scheduling

Parameters Schedule

Run Report Save Parameters

Desformat:

P_DEPTNO:

- Click on the **Schedule** tab. The following screen appears:

Parameters Schedule

Submit

Start

Immediately

At Hour Min AM Month Date Year(YYYY)

Repeat

Only Once

Every hour(s)

Every of of each month

Last of each month on or before the th.

Retry up to time(s) hour(s) after failure

Destination

Content Area

Log File Folder

Result Title

Scroll through this screen and fill in the following information:

- | | |
|--------------------|--|
| Start | Select Immediately . |
| Repeat | Select Every and type 1 in the empty field. This runs the report every hour. |
| Destination | Fill out the following:
Content Area Reports Security Test is already entered for you.
Log File Folder Type Log File . This folder contains information related to the actual running of the report.
Result Title Type Financial information .
Result Folder Type Result Folder . The report output is pushed to this folder.
Expiration Select Permanent from the drop down list.
Overwrite Previous Result By clicking on this, the previous results are overwritten; otherwise, they would be saved. |

4. Click on the **Submit** button.

Return to the **Welcome BJ** folder page and you see that the **Result Folder** has been created. Every hour the report results appear in this folder and BJ can see them each time he clicks on the folder.



5.16 Publishing Report Outside of Oracle Portal

To publish the reports you have registered within the Oracle Portal repository, you are not forced to use Oracle Portal as the deployment mechanism. You can create links on any Web content area or Oracle Portal content area from which users can invoke report requests.

The following steps describe how to implement application-level security outside of the Oracle Portal content area:

1. The SECURITYTNSNAME parameter and the PORTALUSERID parameter must be added to the `rep60_<machinename>.ora` file pointing to the instance where the access control information exists (that is, `PORTALUSERID=<portal_username>/<portal_password>` and `SECURITYTNSNAME=<tnsnames>`). Without this entry, the Oracle Reports Services server does not authenticate the users against the access control information that exists in the Oracle Portal repository.
2. The CGI or servlet key mapping file (`cgicmd.dat`) can utilize the AUTHID command line parameter. You can add the AUTHID parameter to the report request command lines. AUTHID identifies the application user that has been created in Oracle Portal. The `AUTHID=username/password` can be added to any report entry in the key map file to hard code a user name and password. In this case, the user is prompted for application-level security.

If you want the users to be prompted to authenticate every time they run report requests, then add the SHOWAUTH and AUTHTYPE=S entry into the URL, or include the %S argument in the key mapping report entry. For example, an entry in the key map file might look like the following:

```
emp: server=repserver report=emp.rdf userid=scott/tiger@orcl destype=CACHE  
desformat=HTMLCSS %S
```

In the above example, the database user name, password, and connect string (scott/tiger@orcl) are hard coded so the user is only required to enter their user name and password for the application-level security. Meaning the user name and password stored in Oracle Portal are used to identify if they have access to run the selected report. You only see the **User Name** and **Password** fields for the system authentication dialog as shown in the following:

SYSTEM USER AUTHENTICATION

User Name:

Password:

You only see the User Name and Password fields for the system authentication dialog box as in the above. If you omit the USERID parameter from the key mapping entry, then you are also prompted for the database authentication as shown in the following screen:

Database User Authentication

User Name:

Password:

Database:

Another way to enforce database authentication is to add the %D argument to the report key mapping entry. For example:

```
emp: server=repserver report=emp.rdf destype=CACHE desformat=HTMLCSS %S %D
```

You can edit the authentication template file as long as the authentication HTML file contains all comment tags that are marked as `Please do not edit this line.`

3. Specify the `REPORTS60_COOKIE_EXPIRE` value. Each time a user successfully logs into the application to run a report request, the browser is sent an encrypted cookie. When a cookie expires, users must re-authenticate to run subsequent requests. An administrator can define the `REPORTS60_COOKIE_EXPIRE` environment variable on the server. Remember to stop and restart the server for this change to become effective.

The cookie is automatically removed when the client browser is closed. However, in the case where a browser can remain open for an extended period of time, this environment variable helps to control the length of a session. When the Oracle Reports Services CGI executable receives a job request, the amount of time saved in the cookie is compared with the current system time. If the time is longer than the number of minutes defined in the environment variable (by default, 30 minutes), then the cookie is rejected and the user is again requested to identify themselves for authentication.

Refer to [Table 5-1, "Environment Variables for User Authentication"](#) for a list of environment variables that apply to user authentication.

If you want users to authenticate and remain authenticated until the cookie expires, then omit the `AUTHID` parameter or the `%S` argument from the key mapping file.

Configuring Oracle Reports Services Server Clusters

This chapter will show you how to configure Oracle Reports Services servers 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 servers 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 servers to grow as your organization grows.

Before you begin to configure your Oracle Reports Services servers 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 Services Server on Windows NT and UNIX"](#) for more information.

6.1 Clustering Overview

Suppose that you have three machines configured as Oracle Reports Services servers 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 servers in a cluster as described in this overview, see [Section 6.2, "Configuring Oracle Reports Services Servers in a Cluster Example"](#).

You will designate NT-1 as the master, then set the CLUSTERCONFIG parameter to enable this server to recognize NT-2 and SUN-1 as slaves. To simplify this example, the MAXENGINE 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 servers in their own right if they can start engines independently of the master server. Suppose that the MAXENGINE and MINENGINE parameters of the NT-2 Oracle Reports Services configuration are set to three. This means that three engines are dedicated to the NT-2 Oracle Reports Services servers 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 Servers in a Cluster Example

This section provides step-by-step instructions for configuring Oracle Reports Services server 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 `tnsnames.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 you will configure the master using the following settings:

- Edit the master server configuration file to identify the slave servers to the master and to control the number of engines associated with master server.
- Set the parameters in the master server configuration file that defines the following:
 - 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 servers that have been configured for clustering, you specify the master server in the `SERVER` command line argument (that is, `SERVER=NT-1`) along with any other relevant arguments for the thin client executable. The master server assigns incoming jobs to the engines on the slave servers.

If you set the `REPORTS60_REPORTS_SERVER` environment variable to the master server, then you can omit the `SERVER` command line argument. See [Appendix D, "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 Oracle Reports Services Queue Viewer in the Queue Manager. Refer to the Oracle Reports Services 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 server. The TNS service entry name for Oracle Reports Services server 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 restart the Oracle Reports Services server on the NT-1 machine. The NT-1 machine was able to start the slave engines on the two Sun machines, but could not start the slave engines on the NT-2 machine 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.

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 Reports Services Builder in order to be run by Oracle Reports Services. With the 6i 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 Reports Services 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 Reports Services 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). 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. 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>
```

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

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

For more information on this step, refer to [Section 7.3, "Running XML Report Definitions"](#).

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 an 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.

² You can also use XML report definitions with the Oracle Reports Services Runtime and Oracle Reports Services Builder.

- [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:³

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

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.⁴ The `</report>` tag indicates the end of the report definition.

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

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

³ 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 the required tags.

⁴ 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"](#).

7.2.2 Partial Report Definitions

One of the primary 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.⁵ 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 Reports Services 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.

Table 7–1 XML Report Definitions for Building Reports

XML File	RDF File	Description
<code>cond.xml</code>	<code>cond.rdf</code>	<p><code>cond.xml</code> changes:</p> <ul style="list-style-type: none"> ■ The format mask of <code>F_trade_date</code> to <code>MM/DD/RR</code>. ■ The fill colors of <code>F_Mincurrent_pricePersymbol</code> and <code>F_Maxcurrent_pricePersymbol</code>. <p><code>cond.xml</code> adds:</p> <ul style="list-style-type: none"> ■ HTML in the report escapes to be inserted when generating HTML output. <p>Refer to Section 7.2.2.1, "Formatting Modifications Example", for more information.</p>

⁵ It is possible to save the combined RDF file and XML report definition as a new RDF file. This technique is discussed later in this chapter.

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

XML File	RDF File	Description
temp.xml	temp.rdf	<p>temp.xml changes:</p> <ul style="list-style-type: none"> ■ The field labels for F_high_365 and F_low_365. <p>temp.xml adds:</p> <ul style="list-style-type: none"> ■ 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. <p>Refer to Section 7.2.2.2, "Formatting Exception Example", for more information.</p>
sect.xml	sect.rdf	<p>sect.xml adds:</p> <ul style="list-style-type: none"> ■ 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. <p>Refer to Section 7.2.2.3, "Program Unit and Hyperlink Example", for more information</p>
ref.xml	ref.rdf	<p>ref.xml adds:</p> <ul style="list-style-type: none"> ■ 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. <p>Refer to Section 7.2.2.4, "Data Model and Formatting Modifications Example", for more information.</p>

You can apply the XML customizations by running the RDF files with one additional argument. For example:

```

rwr60 userid=scott/tiger report=cond.rdf
      customize=e:\orant\tools\doc60\us\rbr60\cond.xml
    
```


Refer to [Section 7.3, "Running XML Report Definitions"](#) for more information

Take a few moments to run these RDF files with and without the customization file. In the next section, we 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

The following tips are useful when looking at 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 are 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>&nbsp;
              </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

The following tips are useful when looking at 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

The following tips are useful when looking at 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 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.

```
<report name="sect" DIDVersion="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_ssnFormatTrigger">
    <![CDATA[
      function F_ssnFormatTrigger 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

The following tags are useful when looking at 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 is 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 place it inside of the `<field></field>` tag pair. Be aware that if there is both a global and local `<labelAttribute>`, the local one overrides the global one. Refer to [Section 7.5.8, "<field>"](#), for more information

```

<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 Reports Services 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

The following tips are useful when looking at 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 Reports Services Builder and look at the Oracle Data Model view of the Oracle Reports Services Editor:

```
rwbl60 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.

```
<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="corpl.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>
      </group>
    </groupLeft>
  </section>

```



```
<field name="f_SumTOTAL_SALESPerCITY1"
      source="SumTOTAL_SALESPerCITY1"
      label="Sales" font="Arial" fontSize="8"
      formatMask="LNNNGNNNGNNNGNNO00">
  <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="LNNNGNNNGNNNGNNO00">
  <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="LNNNGNNNGNNNGNNO00">
  <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="LNNNGNNNGNNNGNNO00">
  <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="LNNNGNNNGNNNGNNO00">
  <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="LNNNGNNNGNNNGNNO00">
  <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="LNNNGNNNGNNNGNNO00">
    <labelAttribute font="Arial" fontSize="8"
                    fontStyle="bold" textColor="yellow"/>
  </field>
</matrixCol>

```

```

<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. Refer to [Section 7.3.3, "Performing Batch Modifications"](#), for more information about using CUSTOMIZE with RWCON60.

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 Services 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 Services 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 `BeforeForm` or `AfterForm` 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 `BeforeForm` or `AfterForm` triggers of the report:

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

When the report is run, the trigger executes and the specified XML file is 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 `AfterParameterForm` 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 `AfterParameterForm` 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

The following tips are useful when looking at 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="LNNNNNNNNNNNOD00"/>');
  SRW.ADD_DEFINITION(' <field name="F_TOTAL_SALES"
    source="TOTAL_SALES" formatMask="LNNNNNNNNNNNOD00"/>');
  SRW.ADD_DEFINITION(' <field name="F_TOTAL_COST"
    source="TOTAL_COST" formatMask="LNNNNNNNNNNNOD00"/>');
  SRW.ADD_DEFINITION(' <field name="F_SumTOTAL_PROFITPerCITY"
    source="SumTOTAL_PROFITPerCITY"
formatMask="LNNNNNNNNNNNOD00"/>');
  SRW.ADD_DEFINITION(' <field name="F_SumTOTAL_SALESPerCITY"
    source="SumTOTAL_SALESPerCITY"
formatMask="LNNNNNNNNNNNOD00"/>');
  SRW.ADD_DEFINITION(' <field name="F_SumTOTAL_COSTPerCITY"
    source="SumTOTAL_COSTPerCITY"
formatMask="LNNNNNNNNNNNOD00"/>');
  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="LNNNNNNNNNN0"/>');
  SRW.ADD_DEFINITION(' <field name="F_TOTAL_SALES"
    source="TOTAL_SALES" formatMask="LNNNNNNNNNN0"/>');
  SRW.ADD_DEFINITION(' <field name="F_TOTAL_COST"
    source="TOTAL_COST" formatMask="LNNNNNNNNNN0"/>');
  SRW.ADD_DEFINITION(' <field name="F_SumTOTAL_PROFITPerCITY"
    source="SumTOTAL_PROFITPerCITY"
formatMask="LNNNNNNNNNN0"/>');
  SRW.ADD_DEFINITION(' <field name="F_SumTOTAL_SALESPerCITY"
    source="SumTOTAL_SALESPerCITY"
formatMask="LNNNNNNNNNN0"/>');
  SRW.ADD_DEFINITION(' <field name="F_SumTOTAL_COSTPerCITY"
    source="SumTOTAL_COSTPerCITY" formatMask="LNNNNNNNNNN0"/>');
  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" DIDVersion="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" DIDVersion="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" DIDVersion="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 Services Runtime, then the equivalent command line would be:

```
rwr60 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. You could also apply an XML customization file to this report using the CUSTOMIZE argument.

7.3.3 Performing Batch Modifications

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 Reports Services 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=rdf
       source="(inven1.rdf, inven2.rdf, manu.rdf)"
       dtype=rdf
       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 catches most syntax errors and displays 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 generates 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 run 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 Reports Services Builder. In Oracle Reports Services 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 Reports Services 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 Reports Services 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 Reports Services Builder, you use the CUSTOMIZE keyword. For example:

```
rwbl60 userid=username/password@mydb report=emp.rdf
      customize=e:\myreports\emp.xml
```

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

7.4.4 TEXT_IO

If you are using SRW.ADD_DEFINITION to build an XML report definition in memory, 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 ignores 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

Following is the syntax for this tag:

```
<!--  
    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\RBER60.  
    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 ignores 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

Following is the syntax for this tag:

```
<![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_ssnFormatTrigger">  
  <![CDATA[  
    function F_ssnFormatTrigger 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

Following is the syntax for this tag:

```
<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. The valid operators are: <ul style="list-style-type: none"> ▪ 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	<p>Defines whether there are multiple conditions and, if there are, how they should be related.</p> <ul style="list-style-type: none"> ▪ 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

Following is the syntax for this tag:

```
<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>&nbsp;
            </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

Following is the syntax for this tag:

```
<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. The `<dataSource>` tag 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

Following is the syntax for this tag:

```

<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

Following is the syntax for this tag:

```
<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.

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

Attribute	Required or Optional	Description
textColor	Optional	Is the name of the text color to apply when the condition is met.
hide	Optional	Determines whether to hide the field when the condition is met. <ul style="list-style-type: none">▪ 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. The valid styles are: <ul style="list-style-type: none">▪ regular▪ italic▪ bold▪ boldItalic
fontEffect	Optional	Is the effect of the font to be used when the condition is met. The valid values are: <ul style="list-style-type: none">▪ regular▪ strikeout▪ underline▪ strikeoutUnderline

Usage Notes

The following usage notes apply:

- 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>`, and `<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"]
  [fontEffect="regular | strikeout | underline | strikeoutUnderline"]
  [lineColor="color_name | noLine"]
  [fillColor="color_name | noFill"]
  [textColor="color_name"]
  [alignment="start | left | center | right | end"]
  [hyperlink="URL"]
  [linkdest="hyperlink_target"]
  [formatMask="mask"]
/> | >[other_tags]</field>
```

Attributes

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

Table 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 overrides 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.
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. Refer to Section 7.5.15 , "<labelAttribute>", for more information. The label attribute only affects new fields, it does 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. Refer to Section 7.5.22 , "<object>", for more information.
currency	Optional	Is the currency symbol to be used with the field (for example, \$). 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 Reports Services Builder online help system and look for format trigger in the index.

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

Attribute	Required or Optional	Description
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. The valid values are: <ul style="list-style-type: none">▪ regular▪ italic▪ bold▪ boldItalic
fontEffect	Optional	Is the effect of the font to be used for the field contents. The valid values are: <ul style="list-style-type: none">▪ 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. The valid values are: <ul style="list-style-type: none">▪ start▪ left▪ center▪ right▪ end

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

Attribute	Required or Optional	Description
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 Reports Services 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="LNNNGNNNGNNNGNNOD00">
    <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="LNNNGNNNGNNNGNNOD00">
    <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="LNNNGNNNGNNNGNOD00">
  <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

Following is the syntax for this tag:

```
<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

Following is the syntax for this tag:

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

Attributes

The following table describes the attributes of the <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 is 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: <ul style="list-style-type: none">■ 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 places the column in the appropriate group based on where you 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

Following is the syntax for this tag:

```
<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_ssnFormatTrigger">
    <![CDATA[
      function F_ssnFormatTrigger 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

Following is the syntax for this tag:

```
<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="LNNNGNNGNNGNNGNND00">
    <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="LNNNGNNGNNGNNGNND00">
    <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="LNNNGNNGNNGNNGNND00">
    <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 are 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

Following is the syntax for this tag:

```
<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 <groupAbove> 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 <groupLeft> layout.

For more information refer to:

- [Section 7.5.8, "<field>"](#)
- [Section 7.5.12, "<group>"](#)

Syntax

Following is the syntax for this tag:

```
<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 does not change the label of an existing field in the RDF file. 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

Following is the syntax for this tag:

```
<labelAttribute
  [font="font_name"]
  [fontSize="point_size"]
  [fontStyle="regular | italic | bold | boldItalic"]
  [fontEffect="regular | strikethrough | underline | strikethroughUnderline"]
  [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. The valid values are: <ul style="list-style-type: none"> ▪ regular ▪ italic ▪ bold ▪ boldItalic
fontEffect	Optional	Is the effect of the font to be used for the field contents. The valid values are: <ul style="list-style-type: none"> ▪ 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. The valid values are: <ul style="list-style-type: none"> ▪ 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

Following is the syntax for this tag:

```
<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 is 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="LNNNGNNNGNNNGNND00">
          <labelAttribute font="Arial" fontSize="8"
            fontStyle="bold" textColor="black"/>
        </field>
      </group>
    </matrix>
  </section>
</layout>
```

```
<field name="f_SumTOTAL_COSTPerQUARTER"
      source="SumTOTAL_COSTPerQUARTER"
      label="Costs: " font="Arial" fontSize="8" fontStyle="bold"
      formatMask="LNNNGNNNGNNNGN0D00">
  <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="LNNNGNNNGNNNGN0D00">
  <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="LNNNGNNNGNNNGN0D00">
    <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="LNNNGNNNGNNNGN0D00">
    <labelAttribute font="Arial" fontSize="8"
                   fontStyle="bold" textColor="yellow"/>
  </field>
</matrixCol>
```

```

    <field name="f_SumTOTAL_PROFITPerCITY"
          source="SumTOTAL_PROFITPerCITY"
          label="Sales: " font="Arial" fontSize="8" fontStyle="bold"
          textColor="yellow" formatMask="LNNNGNNGNNGNNGNND00">
      <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="LNNNGNNGNNGNNGNND00"/>
  <field name="f_total_cost" source="total_cost" label="Total Cost"
        font="Arial" fontSize="8" lineColor="noLine"
        formatMask="LNNNGNNGNNGNNGNND00"/>
  <field name="f_total_profit" source="total_profit" label="Total Profit"
        font="Arial" fontSize="8" lineColor="noLine"
        formatMask="LNNNGNNGNNGNNGNND00"/>
</matrixCell>
</matrix>
</section>
</layout>

```

7.5.17 <link>

Description

The <link> tag defines a link between data sources in the data model. The <link> tag 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

Following is the syntax for this tag:

```
<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	<p>Is a SQL operator that defines the relationship between parent column and child column. Condition can have the following values:</p> <ul style="list-style-type: none"> ▪ 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	<p>Is the type of SQL clause that relates the parent group to the child query. The default is a WHERE clause.</p>

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

Following is the syntax for this tag:

```
<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="LNNNGNNGNNGNNGN0D00">
      <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="LNNNGNNGNNGNNGN0D00">
      <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="LNNNGNNGNNGNNGN0D00">
      <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="LNNNGNNGNNGNNGN0D00">
    <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="LNNNGNNGNNGNNGN0D00">
    <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="LNNNGNNGNNGNNGN0D00">
    <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="LNNNGNNGNNGNNGN0D00"/>
  <field name="f_total_cost" source="total_cost" label="Total Cost"
    font="Arial" fontSize="8" lineColor="noLine"
    formatMask="LNNNGNNGNNGNNGN0D00"/>
</matrixCell>
```

```

    <field name="f_total_profit" source="total_profit" label="Total Profit"
          font="Arial" fontSize="8" lineColor="noLine"
          formatMask="LNNNGNNNGNNNGNNOD00"/>
  </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

Following is the syntax for this tag:

```

<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="LNNNGNNNGNNNGNNOD00"/>
  <field name="f_total_cost" source="total_cost" label="Total Cost"
        font="Arial" fontSize="8" lineColor="noLine"
        formatMask="LNNNGNNNGNNNGNNOD00"/>
  <field name="f_total_profit" source="total_profit" label="Total Profit"
        font="Arial" fontSize="8" lineColor="noLine"
        formatMask="LNNNGNNNGNNNGNNOD00"/>
</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

Following is the syntax for this tag:

```
<matrixCol>
  master_group_content
</matrixCol>
```

Example

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

```
<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="LNNNGNNGNNGNNGN0D00">
    <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="LNNNGNNGNNGNNGN0D00">
    <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="LNNNGNNNGNNNGNNOD00">
  <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

Following is the syntax for this tag:

```
<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

Following is the syntax for this tag:

```
<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: <ul style="list-style-type: none">▪ 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>&nbsp;
            </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

Following is the syntax for this tag:

```
<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_ssnFormatTrigger">
  <![CDATA[
    function F_ssnFormatTrigger 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

Following is the syntax for this tag:

```

<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>&nbsp;
            </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

Following is the syntax for this tag:

```
<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)	<ul style="list-style-type: none">▪ xmlTag▪ xmlAttribute▪ xmlSuppress▪ prologType▪ prolog▪ beforeReportValue▪ beforeReportType▪ afterReportValue▪ afterReportType▪ beforePageValue▪ beforePageType▪ afterPageValue▪ afterPageType▪ beforeFormValue▪ beforeFormType▪ afterFormValue▪ afterFormType▪ pageNavigationControlValue▪ pageNavigationControlType
Group object (REP_GROUP)	<ul style="list-style-type: none">▪ xmlTag▪ xmlAttribute▪ outerXMLTag▪ outerXMLAttribute▪ xmlSuppress
Column object (REP_COL_MAP)	<ul style="list-style-type: none">▪ 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>&nbsp;
            </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

Following is the syntax for this tag:

```
<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. 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\RBER60.
  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>&nbsp;
            </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 Services 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

Following is the syntax for this tag:

```
<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

Following is the syntax for this tag:

```
<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

Following is the syntax for this tag:

```
<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.
function	Optional	Is the mathematical operation to be applied to produce the summary values: <ul style="list-style-type: none">■ 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.

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

Attribute	Required or Optional	Description
compute	Optional	<p>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.</p> <p>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.</p>
reset	Optional	<p>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.</p> <p>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.</p>
productOrder	Optional	<p>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. The productOrder attribute 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.</p>

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

Attribute	Required or Optional	Description
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	<ul style="list-style-type: none"> ■ 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

Following is the syntax for this tag:

```
<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

[Appendix A, "Controlling User Access to Reports by Defining Calendars"](#)

[Appendix B, "RWCLI60 Command Line Arguments"](#)

[Appendix C, "Oracle Reports Services Configuration Parameters"](#)

[Appendix D, "Environment Variables"](#)

[Appendix E, "Database Connection Strings"](#)

[Appendix F, "Migrating from Web Cartridge to CGI"](#)

[Appendix G, "Troubleshooting"](#)

Controlling User Access to Reports by Defining Calendars

As discussed in [Chapter 5, "Oracle Reports Services Security with Oracle Portal"](#) access control enables you to restrict user access to reports that are run with Oracle Reports Services. Oracle Reports Services uses Oracle Portal to perform a security check that ensures that users have the necessary privileges to run reports on restricted Oracle Reports Services servers and printers. Defining calendars is an optional step that allows you to further restrict access to report definition files (RDFs), servers, and printers by determining when they can and cannot be accessed.

A.1 Creating Availability Calendars

An availability calendar determines when RDFs, Oracle Reports Services servers, and printers are available for processing. Availability calendars are not necessary if the RDFs, Oracle Reports Services servers, and printers are always available for processing.

You can create two types of availability calendars:

- Simple

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.).

- Combined

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 an RDF, Oracle Reports Services servers, or printer. If your production environment requires more than one availability rule, then you need to combine availability calendars.

A.2 Availability Calendar Example

In this example, you need to create a Production calendar that determines the availability for every day of the week, days with scheduled maintenance, and holidays. To do this Production calendar, you need to create the following availability calendars:

- A simple Daily calendar with an availability period of every Sunday through Saturday from 12:00 a.m. to 10:00 p.m.
- A simple Maintenance calendar with an availability period of every Saturday from 3:00 p.m. to 10:00 p.m.
- A simple Christmas calendar with an availability period starting on December 25th at 12:00 a.m. and ending on December 26th at 12:00 a.m.
- A 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.

A.2.1 Creating a Daily Calendar

Create a Daily calendar with an availability period of Sunday through Saturday from 12:00 a.m. to 10:00 p.m. by doing the following:

1. Access Oracle Portal and log on. You must have RW_ADMINISTRATOR and DBA privileges to access Oracle Reports Services security wizards.
2. On the Oracle Portal home page, click the **Administer** tab.
3. On the **Administer** page under the **Oracle Reports Security** portlet, click **Oracle Reports Security Settings**.
4. On the **Oracle Reports Security Settings** page under the **Reports Calendar Access** portlet, click on **Create Simple Calendar Access** to create a simple calendar.
5. On the **Create Simple Availability Calendar** page, type **Daily** in the **Calendar Name** field. If the Daily calendar already exists, then append your initials to it (for example, DailyAA).
6. Click on the **Next** button to continue.

7. This screen is where the **Date/Time Availability** is set. Under **Duration** specify today's date as the start month, date, and year, and 12:00 a.m. as the start time. Specify today's date as the end month, date, and year, and 10:00 p.m. as the end time.
8. Choose **Daily** as the **Repeat** option. This repeats the **Duration** pattern every day. For example, if the start date is Monday, January 4, 2000, then this pattern repeats every day starting on this date until the pattern is completed.
9. Click on the **Next** button to continue.
10. Optionally, on the **Show 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.
11. Click on the **Finish** button.
12. The **Manage Component** screen appears. Click on the **Close** button.

A.2.2 Creating the Maintenance Calendar

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 **Oracle Reports Security** page under the **Reports Calendar Access** portlet, click **Create Reports Simple Calendar Access** option to create a simple calendar.
2. On the **Create Simple Availability Calendar** page, type **Maintenance** in the **Calendar Name** field. If the Maintenance calendar already exists, then append your initials to it (for example, MaintenanceAA).
3. Click on the **Next** button to continue.

- Define the following for **Date/Time Availability**:

Table A-1 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.
End	Specify the same date defined as the start date, and time ending at 10:00 p.m.
Repeat	Choose Weekly .

- Click on the **Next** button to continue.
- Optionally, on the **Show Simple Availability Calendar Summary** page, click **Show Calendar** to view a visual representation of the Maintenance calendar. Green indicates availability. Close the calendar when you are finished reviewing it.
- Click on the **Finish** button.
- The **Manage Component** screen appears. Click on the **Close** button.

A.2.3 Creating the Christmas Calendar

Create a Christmas calendar with an availability period of every December 25th from 12:00 a.m. to December 26th 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.

- From the **Oracle Reports Security** page under the **Reports Calendar Access** portlet, click **Create Reports Simple Calendar Access** option to create a calendar.
- On the **Create Simple Availability Calendar** page, type **Christmas** in the **Calendar Name** field. If the Christmas calendar already exists, then append your initials to it (for example, MaintenanceAA).
- Click on the **Next** button to continue.

4. Define the following for **Date/Time Availability**:

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

5. Click on the **Next** button to continue.
6. Optionally, on the **Show Simple Availability Calendar Summary** page, click **Show Calendar** to view a visual representation of the Christmas calendar. Green indicates availability. Close the calendar when you are finished reviewing it.
7. Click on the **Finish** button.
8. The **Manage Component** screen appears. Click on the **Close** button.

A.2.4 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 **Oracle Reports Security** page, click the **Create Reports Combined Calendar Access** to create the calendar that combines the three calendars you created into one.
2. On the **Create Combined Availability Calendar** page, type **Production** in the **Calendar Name** field. If the Production calendar already exists, then append your initials to it (for example, ProductionAA).
3. Click on the **Next** button to continue.
4. On the **Select Availability Calendars** page, **ctrl-click** on the **Daily**, **Maintenance**, and **Christmas** calendars in the **Availability Calendars** list box.
5. Click on the right arrow to move the selected calendars to the **Selected Availability Calendars** list box, or click on the double right arrow to select all available calendars.
6. Click on the **Next** button to continue.

7. On the **Exclude Availability Calendars** page, **ctrl-click** on the **Maintenance** and **Christmas** calendars in the **Availability Calendars** list box.
8. Click on the right arrow 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.
9. Click on the **Next** button to continue.
10. On the **Show 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 a good practice to check the combined calendar at this point. You can verify that the calendars you prohibited processing on are excluded during the period specified. Scroll to December to ensure that December 25th is excluded from processing. Choose the **Day** option and scroll to a Saturday to ensure that processing is unavailable from 3 p.m.

11. Click on the **Finish** button.
12. On the **Manage Component** page, click on the **Close** button.

You have now successfully created both Simple and Combined calendars. You can now use these calendars to further restrict access to RDFs, Oracle Reports Services servers, and printers. Refer to [Chapter 5, "Oracle Reports Services Security with Oracle Portal"](#) for more information about restricting RDFs, Oracle Reports Services servers, and printers.

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.

B.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
```

B.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 Services Runtime searches first for a file with extension REP, then extension RDF, then XML, and then no extension. Oracle Reports Services 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 CGI command SHOWAUTH and AUTHTYPE=S in the report URL, or include the %D argument to the key mapping entry in the cgicmd.dat (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 E, "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=]termtyp

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 Services 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 the Oracle Reports Services server 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 Reports Services 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 Reports Services 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: $(8.5 * 80)/80 \times (11 * 20)/66 = 8.5 \times 3.33$.

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 Reports Services 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 Reports Services 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 Reports Services Builder Time and ORACLE Time.
- Time is the amount of time spent in Oracle Reports Services 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 Reports Services Builder time includes the database time because the database is included in the Oracle Reports Services Builder 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 Reports Services 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 Services 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 Reports Services 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 Reports Services 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 Reports Services 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 Reports Services 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 Reports Services 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 Reports Services 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 Services 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 Services 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
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:

<code>server</code>	is the Express Server string (for example, <code>ncacn_ip_tcp:olap2-pc/sl=x/st=x/ct=x/sv=x/</code>). See below for more details on the server string.
<code>domain</code>	is the Express Server domain.
<code>user</code>	is the user ID to log on to the Express Server.
<code>password</code>	is the password for the user ID.
<code>ramuser</code>	is the user ID to log into the RDBMS.
<code>rampassword</code>	is the password for the RDBMS.
<code>ramexpressid</code>	is the Oracle Sales Analyzer database user ID. This is required for Oracle Sales Analyzer databases only.
<code>ramserverscript</code>	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 <code>\\ServerName\ShareName\</code> followed by any subfolders or files.
<code>rammasterdb</code>	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 <code>ServerDBPath</code> for Express Server. You can check the <code>ServerDBPath</code> in the File I/O tab of the Express Configuration Manager dialog box.
<code>ramconnecttype</code>	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 server. User authentication ensures that the users making report requests have access privileges to run the requested report. When users successfully log on, their browser is sent an encrypted cookie that authenticates them to the secured Oracle Reports Services server registered in Oracle Portal. By default, the cookie expires after 30 minutes. When a cookie expires, subsequent requests (that is, ones sent to a secured Oracle Reports Services server) must be re-authenticated.

You can use the `REPORTS60_COOKIE_EXPIRE` environment variable to change the expiration time of the authentication cookie. See [Appendix D, "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 CGI command `SHOWAUTH` and `AUTHTYPE=S` in the report URL, or include the `%S` argument to the key mapping entry in the `cgicmd.dat` (CGI) file.

Syntax `[AUTHID=]username/password`

Values Any valid user name and password created in Oracle Portal. See your DBA to create new users accounts in Oracle Portal.

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.

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 CACHESIZE (MB) setting.

Parameter	Description
CLUSTERCONFIG	<p>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:</p> <pre>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>)"</pre> <p>Note: Each slave definition must be enclosed in parentheses. See Chapter 6, "Configuring Oracle Reports Services Server Clusters" for detailed instructions.</p>
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	<p>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:</p> <pre>FAILNOTEFILE="C:\ORACLE_HOME\Report60\failnote.dat"</pre>
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 Services 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 Oracle Reports Services 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	<p>LOGOPTION is the type of log information you want inserted into the log file. The options are alljob, failedjob, and succeededjob. For example:</p> <pre>LOGOPTION="alljob"</pre>

Parameter	Description
MAILPROFILE	<p>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:</p> <p>MAILPROFILE="mailprofileid/password"</p> <p>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.</p> <p>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.</p>
MAXCONNECT	<p>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.</p>
MAXENGINE	<p>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.</p> <p>Note: You can set this parameter from the Oracle Reports Services Queue Manager. Open the Oracle Reports Services Queue Manager and log on as the administrator. Choose Queue → Properties, and then change the Simultaneous running engines Max setting.</p>
MAXIDLE	<p>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.</p> <p>Note: You can set this parameter from the Oracle Reports Services Queue Manager. Open the Oracle Reports Services Queue Manager and log on as the administrator. Choose Queue → Properties, and then change the MAXIDLE time (minutes) before engine shutdown setting.</p>

Parameter	Description
MINENGINE	<p>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.</p> <p>Note: You can set this parameter from the Oracle Reports Services Queue Manager. Open the Oracle Reports Services Queue Manager and log on as the administrator. Choose Queue → Properties, and then change the change the Simultaneous running engines Min setting.</p>
PERSISTFILE	<p>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:</p> <p>PERSISTFILE="C:\ORACLE_HOME\Report60\repserver.dat"</p>
REPOSITORYCONN	<p>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.</p> <p>To create a queue activity table in your database, you must run <code>rw_server.sql</code> script. For example:</p> <p>REPOSITORYCONN="repserver_schema/password@mydb"</p>
SECURITY	<p>SECURITY is the security level (0, 1, 2, or 3) for accessing cached output files through the Oracle Reports Services 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.</p> <p>The default value is 1.</p>

Parameter	Description
SECURITYTNSNAME	<p>SECURITYTNSNAME is the TNS name of the Oracle Portal database that is used for authenticating users to Oracle Reports Services. Oracle Reports Services uses Oracle Portal to perform a security check and to ensure that users have access privileges to run the report to the restricted Oracle Reports Services servers and, if requested, output to a restricted printer.</p> <p>When the SECURITYTNSNAME parameter is set, you must add information about Oracle Reports Services servers, printers, and reports in Oracle Portal to process report requests through Oracle Reports Services. For example:</p> <p>SECURITYTNSNAME="sec_db"</p> <p>See Chapter 5, "Oracle Reports Services Security with Oracle Portal" for more information.</p>
SOURCEDIR	<p>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:</p> <p>SOURCEDIR="C:\my_reports"</p>
SUCCNOTEFILE	<p>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:</p> <p>SUCCNOTEFILE="C:\ORACLE_HOME\REPORT60\succnote.dat"</p>
TEMPDIR	<p>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</p> <p>TEMPDIR="C:\ORACLE_HOME\Report60\temp"</p>

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 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 server to run report requests. When users successfully log on, their browser is sent an encrypted cookie. When a cookie expires, subsequent requests (that is, ones that are sent to secured Oracle Reports Services servers), 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 <code>dbauth.htm</code> .
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 <code>reports6.0</code> .

Variable	Description
REPORTS60_CGIDIAGBODYTAGS	For the Oracle Reports Services server 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 Services server 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_CGIELP	<p>For the Oracle Reports Services server CGI, defines URL and URI of the RWCGI60 help file, which is navigated to when RWCGI60 is invoked with the empty request:</p> <p><code>http://your_webserver/rwcgi60?.</code></p> <p>For example., setting it to <code>http://www.yahoo.com</code> goes to that URL; setting it to <code>myhelpfile.htm</code> displays the file:</p> <p><code>http://your_webserver/myhelpfile.htm</code></p> <p>If this parameter is not defined, then a default help screen is displayed.</p>
REPORTS60_CGIMAP	<p>For the Oracle Reports Services server CGI, defines fully qualified file name and location of the RWCGI60 map file if map file configuration is used. For example:</p> <p><code>C:\ORANT\REPORT60\cgicmd.dat</code></p>
REPORTS60_CGINODIAG	<p>For the Oracle Reports Services server 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:</p> <p><code>http://your_webserver/rwcgi60/help?</code></p>
REPORTS60_REPORTS_SERVER	Specifies the default Oracle Reports Services server for 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.

Variable	Description
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 restricted Oracle Reports Services server.

Database Connection Strings

This appendix lists typical database connection strings that you or users can use when specifying report requests using the CGI or servlet. A database connection string is the value used in the USERID command line argument to connect to the database.

See [Appendix B, "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. If the password cannot be found, then Oracle Reports Services returns the database authentication form.	None. 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. If there is no local database, then Oracle Reports Services returns the database authentication form.	None. 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. 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. If the password cannot be found in the table, then Oracle Reports Services returns the database authentication form.	Types the Oracle user name and password. None. 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.

Migrating from Web Cartridge to CGI

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

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

F.1 Benefits of Migrating to CGI

CGI is a component of the Web HTTP protocol. It is a standard, platform-independent way to dynamically communicate with the Oracle Reports Services 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.

F.2 Steps for Migrating to CGI

Migrating to CGI involves the following steps:

1. Installing the software.
2. Configuring OAS
3. Configuring the CGI
4. Setting environment variables (optional)
5. Renaming the map files (optional)
6. Running a report using the 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 Services server using the Web Cartridge.

F.2.1 Step 1. Installing the Software

You need to do the following to install the software:

1. Install the Oracle Reports Services Developer Thin Client, if you have not already done so.
2. Ensure that a TNSnames service entry exists for the Oracle Reports Services in the `tnsnames.ora` file that is used by the CGI to communicate with the Oracle Reports Services server.
 - If OAS is installed on a different machine than the Oracle Reports Services, then check the `tnsnames.ora` file to ensure that a TNSnames service entry exists for the Oracle Reports Services server in the OAS `ORACLE_HOME`.
 - If OAS release 4.0.8 is installed on the same machine as the Oracle Reports Services, but in a different Oracle home, then you will need to add a TNSnames service entry for the Oracle Reports Services server in the `tnsnames.ora` file in the OAS `ORACLE_HOME`.
 - If OAS release 4.0 (previous to OAS 4.0.8) is installed on the same machine as the Oracle Reports Services and is in the same `ORACLE_HOME`, then no additional TNSnames entries are required.

F.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

F.2.3 Step 3. Configuring the CGI

To configure the CGI, you do the following:

1. On Windows, copy the `rwsgi60.exe` file (located in the `ORACLE_HOME\BIN` directory) to your CGI-BIN directory. On UNIX, copy the `rwsgi60` 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 Services are in different home directories, then you also need to copy the Oracle Reports Services home into the OAS home.

On Windows, if the Oracle Reports Services 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 Services 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/dvsysam.htm
```

F.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 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/rrows?showenv
```

F.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 CGI environment variable.

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

F.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
```

F.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 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 CGI.

F.2.6 Step 6. Running a Report Using the CGI URL

You need to test that you have successfully migrated to CGI. You can test the configuration so ensure that it can communicate with the Oracle Reports Services server. The URL for CGI is different than the URL for the Web cartridge. Run a test using the following CGI URL. For this example, it is assumed that the `REPORTS60_REPORT_SERVER` environment variable was set to point to a default Oracle Reports Services 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 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 CGI to communicate with the Oracle Reports Services server.

F.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 Services server for access control using Oracle Portal, then you will need to change the Web gateway value in the Oracle Reports Services server access control, which was created for the Oracle Reports Services server.

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 <code>tnsnames.ora</code> file and Oracle Reports Services cannot resolve the TNSname. 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 <code>uiprint.txt</code> file.

Problem Description	Probable Cause and Solution
<p>Upon starting Oracle Reports Services, you get server specific error 186.</p>	<p>Typically this indicates a problem in <code>tnsnames.ora</code> or <code>sqlnet.ora</code>. Check the entry for Oracle Reports Services in <code>tnsnames.ora</code>. A typical entry should look something like the following:</p> <pre>repserver.world = (ADDRESS=(PROTOCOL=tcp) (HOST=144.25.87.182)(PORT=1951))</pre> <p>In this example <code>.world</code> is appended to the name because it is the domain specified in the <code>sqlnet.ora</code> file. If the <code>NAMES.DEFAULT_DOMAIN</code> setting is not defined in the <code>sqlnet.ora</code>, then omit <code>.world</code> from the name of the server instance.</p> <p>If your <code>tnsnames.ora</code> file appears to be correct, then check your <code>sqlnet.ora</code> file. Good default settings to use in this file are:</p> <pre>TRACE_LEVEL_CLIENT=OFF names.directory_path = (TNSNAMES) names.default_domain = world name.default_zone = world</pre> <p>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.</p>
<p>Error reported when opening the report.</p>	<p>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 <code>your_report.rdf.txt</code>. Alternatively, use a DOS window for file manipulation.</p> <p>Alternatively, ensure the report is located in the path defined by the <code>REPORTS60_PATH</code> environment variable.</p>

Problem Description	Probable Cause and Solution
Problems running Oracle Reports Services as a Windows NT Service.	<p>If you install Oracle Reports Services service to run under a user other than SYSTEM, then ensure the user account:</p> <ul style="list-style-type: none"> ■ 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. <p>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 Reports Services Builder online help.</p>
ops\$ account is not working.	<p>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.</p>
Database roles not working as expected.	<p>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.</p>

Problem Description	Probable Cause and Solution
URL mapping is not working.	<p>Ensure you have a valid key mapping file. It must be named <code>cgicmd.dat</code> (for the Oracle Reports Services server CGI or servlet) in the <code>REPORT60</code> directory, or named according to the value set in the <code>REPORTS60_CGIMAP</code> environment variable.</p> <p>To ensure the key mapping file can be found, first try the following (a CGI example) and verify that your key entry has been correctly parsed in the resulting page:</p> <p><code>http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe/showmap?</code></p> <p>Then try, running the report using the key map entry, where <code>your_key</code> is a valid key entry in the key mapping file:</p> <p><code>http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe?your_key</code></p>
Cannot shutdown the queue from the Oracle Reports Services Queue Manager.	<p>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 Services 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.</p>
Cannot run Oracle Reports Services as an NT Service under LocalSystem.	<p>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 <code>sqlnet.ora</code> file, otherwise the server cannot be accessed:</p> <pre>sqlnet.authentication_services=(NONE)</pre>
Problems finding files.	<p>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:</p> <pre>\\SALES\DOCUMENTS\REPORTS)</pre> <p>This applies to Oracle Reports Services parameters, CGI and servlet command mappings, and each hard-coded path name in each report being run.</p>

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 <code>your_report.rdf.txt</code> . 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 Oracle Portal.	<p>In Oracle Portal, verify the access controls that you defined for the printer, Oracle Reports Services server, and report definition file.</p> <p>Check for the following:</p> <ul style="list-style-type: none"> ■ 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 server and optionally, the printer has been created. ■ Users who require access to the report definition files, servers, and printer have been given access to them. <p>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 server.</p>

Problem Description	Probable Cause and Solution
<p>Reports are not running when the URL is requested.</p>	<p>Check for the following:</p> <ul style="list-style-type: none"> ■ 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 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: <pre>http://your_webserver/your_virtual_cgi_dir/rwccgm60.exe or http://your_webserver/rwows</pre> For UNIX, type: <pre>http://your_webserver/your_virtual_cgi_dir/rwccgi60 or http://your_webserver/rwows</pre> A help page should appear. If it does not, then check the mapping of <code>your_virtual_cgi_dir</code> (usually called <code>cgi-bin</code>) 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 <code>REPORTS60_CGINODIAG</code> (for CGI or servlet) environment variable is not defined, otherwise all diagnostic output is disabled. Test this by typing one of the following: <pre>http://your_webserver/your_virtual_cgi_dir/rwccgi60.exe/ showenv? http://your_webserver/rwows/showenv?</pre> This also allows you to view the other parameters or environment variables.

Problem Description	Probable Cause and Solution
	<ul style="list-style-type: none"> <li data-bbox="648 265 1338 348">■ 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? <li data-bbox="648 423 1338 506">■ Try running a simple report to your browser, by typing one of the following: http://your_webserver/your_virtual_cgi_dir/rwcgi60.exe?server=your_repserver+report=your_report.rdf+userid=scott/tiger@mydb+desformat=html http://your_webserver/rwows?server=your_repserver+report=your_report.rdf+userid=scott/tiger@my_db+desformat=html <p data-bbox="648 661 1233 689">If the report does not display, then check to ensure that:</p> <ul style="list-style-type: none"> <li data-bbox="648 701 1338 808">■ Your_report.rdf runs correctly from Oracle Reports Services Builder or Oracle Reports Services Runtime Your_report.rdf is located in a directory specified under REPORTS60_PATH. <li data-bbox="648 821 1129 848">■ The database connection string is correct. <li data-bbox="648 861 1338 968">■ The Oracle Reports Services 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. <li data-bbox="648 980 1338 1088">■ 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 Services server. Contact your Oracle Reports Services system administrator. <p data-bbox="648 1100 1338 1156">Remember that the Oracle Reports Services server must have access to the report and any external files used by the report.</p> <p data-bbox="648 1168 1338 1387">When sending a report to the Oracle Reports Services 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 Services 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.</p>
Report does not output to the printer.	You might have access privileges to run a report to restricted Oracle Reports Services 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.	<p data-bbox="601 262 1268 366">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.</p> <p data-bbox="601 383 1268 461">If you trying to run your report to a restricted Oracle Reports Services server, then the Web Gateway URL defined in the Server Access in Oracle Portal might be incorrect.</p> <p data-bbox="601 479 1268 635">In Oracle Portal, click Administrator from the Oracle Portal 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.</p> <p data-bbox="601 652 1268 730">Note: Only users with Oracle Reports Services system administrator privileges can access Oracle Reports Services Security in Oracle Portal.</p>

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 the Oracle Reports Services 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 server, 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 server) must re-authenticate.

CSS (Cascading Style Sheets)

HTML with CSS allows developers to control the style and layout of multiple Web pages all at once. A style sheet works like 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.

Oracle9i Application Server

Oracle9i 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. Oracle9i 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 the Oracle Reports Services server gets the corresponding password from the cookie and runs the report.

Port

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

Push delivery

The delivery of information on the Web that is initiated by the server rather than by a client request. Oracle Reports Services can push reports to an Oracle Portal site by scheduling the report request to run automatically on a secured Oracle Reports Services server. The end user clicks the link on the Oracle Portal site to view the report.

Oracle Portal

Oracle Portal is an HTML-based development tool for building scalable, secure, extensible HTML applications and Web sites. Oracle Reports Services uses Oracle Portal 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.

Oracle Portal component

A PL/SQL stored procedure created by an Oracle Portal 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.

Oracle Reports Services Queue Manager

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

Oracle Reports Services Launcher

An application that utilizes the functionality provided by the Oracle Reports Services ActiveX control, such as 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 Services Server Servlet

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

Oracle Reports Services Server CGI

An interface between a CGI-aware Web server and Oracle Reports Services 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 Oracle9i 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).

Web Server

A server process (`httpd` daemon) running at a Web site which sends out Web pages in response to `http` requests from remote Web browsers.

Index

Symbols

<!-- --> XML tag reference, 7-30
<![CDATA[]]> XML tag reference, 7-31
<condition> XML tag reference, 7-32
<customize> XML tag reference, 7-34
<data> XML tag reference, 7-36
<dataSource> XML tag reference, 7-37
<exception> XML tag reference, 7-39
<field> XML tag reference, 7-41
<formLike> XML tag reference, 7-46
<formula> XML tag reference, 7-47
<function> XML tag reference, 7-50
<group> XML tag reference, 7-51
<groupAbove> XML tag reference, 7-53
<groupLeft> XML tag reference, 7-54
<labelAttribute> XML tag reference, 7-56
<layout> XML tag reference, 7-58
<link> XML tag reference, 7-61
<matrix> XML tag reference, 7-64
<matrixCell> XML tag reference, 7-67
<matrixCol> XML tag reference, 7-68
<matrixRow> XML tag reference, 7-69
<object> XML tag reference, 7-70
<programUnits> XML tag reference, 7-72
<properties> XML tag reference, 7-73
<property> XML tag reference, 7-75
<report> XML tag reference, 7-78
<section> XML tag reference, 7-80
<select> XML tag reference, 7-82
<summary> XML tag reference, 7-83
<tabular> XML tag reference, 7-88

A

access control
 availability calendars, A-1
ActiveX request method, 4-2
adding
 another slave server to the master, 6-8
Apache, 3-9, 3-10
Apache server See Oracle HTTP Server powered by
 Apache
applying in XML
 multiple report definitions, 7-20
 report definition at runtime, 7-19
 report definition in PL/SQL, 7-21
 report definition stored in a file, 7-21
 report definition stored in memory, 7-21
architecture
 Oracle Reports Services, 1-2
 Oracle Reports Services tier, 1-2
 database, 1-2
 thin client, 1-2
 Web server, 1-2
 Web server configurations, 1-3
authentication cookie
 expiring, B-29
availability calendar, A-1

B

batch

- modifications to reports, 7-25
- reporting
 - from an Oracle Portal site, 4-9

C

cache

- size, C-1

CACHESIZE parameter, 4-3

CLUSTERCONFIG parameter, 6-2, 6-5

clustering

configuring

- master server, 6-5
- Oracle Reports Services, 6-3
- Oracle Reports Services servers, 6-3

enabling communication between master and slave, 6-4

overview, 6-2

resubmitting, 6-7

running reports, 6-7

combined availability calendar, A-5

command line arguments, B-1

AUTHID, B-29

CURRENCY, 4-3

CUSTOMIZE, 7-3, 7-19, 7-20, 7-25

DECIMAL, 4-3

DESFORMAT, 4-3

DESTYPE, 4-3, B-4

mapping URL parameter, 4-4

MODE, 4-3

ORIENTATION, 4-3

PAGESIZE, 4-3

PARAMFORM, 4-3

REPORT, 4-3, 7-3, 7-19

RWCLI60, 4-6, 4-8, B-1

SCHEDULE, 6-7

SERVER, 3-16, 3-17, 4-2, 4-6, 4-8, 6-7, B-22, D-2, F-5

THOUSANDS, 4-3

TOLERANCE, 4-3

USERID, 4-3, B-2, E-1

commands

line arguments, 1-5

Oracle Reports Services Runtime, 4-2

READONLY, B-9, B-13

RWCLI60, 4-1, 4-2, 7-20, 7-25

RWRUN60, 7-10, 7-21, 7-25, B-3, B-17

SCHEDULE, B-23

setenv, 3-12

SHOWAUTH, B-2, B-29

SRW.RUN.REPORT, 4-2

concepts, 1-1

configuring

master server clustering, 6-5

Oracle Reports Services, 1-7

Oracle Reports Services server

modifying, 3-17

UNIX with environment variables, 3-14

Windows NT with environment

variables, 3-12

with environment variables, 3-12

Oracle Reports Services server CGI, 3-8, 3-10

Oracle Reports Services server clustering, 6-1, 6-3

Oracle Reports Services server servlet, 3-4

with JSDK, 3-5

with JServ, 3-7

Web server, 3-9

creating

a service entry for Oracle Reports Services server, 3-11

availability calendar, A-1

combined availability calendar, A-5

daily calendar, A-2

maintenance calendar, A-3

XML

report definition required tags, 7-4

report definitions, 7-2, 7-3

CURRENCY command line argument, 4-3

CUSTOMIZE

command line argument, 7-3

keyword, 7-29

customizing

overview, 7-2

reports at runtime, 7-1

XML report definition, 7-3

D

daily calendar, A-2
database tier, Oracle Reports Services, 1-2
debugging
 tracing options, 7-26
 XML report definitions, 7-26
DECIMAL command line argument, 4-3
default printer, set access, 3-3, 3-15
DESFORMAT
 command line argument, 4-3
DESTYPE
 command line argument, 4-3
directories
 ORACLE_HOME/guicommon6/tk60/ADMIN,
 3-15
 ORACLE_HOMEREPORT60, 4-5
 ORACLE_HOMEREPORT60SERVER, 3-14
duplicate job detection
 multiple output destinations, 4-3
 Oracle Reports Services handling, 4-3, B-23

E

ENGLIFE parameter, 6-5
environment variables
 configuration, 3-15
 REPORTS_PATH, 3-14
 TNS_ADMIN, 3-14
 REPORTS_REPORTS_SERVER, 3-17
 REPORTS60_CGIMAP, 4-5
 REPORTS60_COOKIE_EXPIRE, 3-15
 REPORTS60_DB_AUTH, 3-16
 REPORTS60_ENCRYPTION_KEY, 3-16
 REPORTS60_PATH, 3-13, 3-14, 4-7
 REPORTS60_REPORT_SERVER, F-5
 REPORTS60_REPORTS_SERVER, 3-12, 3-16,
 4-6, 4-8, 6-7
 REPORTS60_SSLPORT, 3-16
 REPORTS60_SYS_AUTH, 3-16
 TNS_ADMIN, 3-13, 3-14
examples
 <!-- --> XML tag reference, 7-30
 <![CDATA[]]> XML tag reference, 7-31
 <condition> XML tag reference, 7-34
 <customize> XML tag reference, 7-35

 <data> XML tag reference, 7-36
 <dataSource> XML tag reference, 7-38
 <exception> XML tag reference, 7-41
 <field> XML tag reference, 7-45
 <formLike> XML tag reference, 7-49
 <formula> XML tag reference, 7-47
 <function> XML tag reference, 7-50
 <group> XML tag reference, 7-52
 <groupAbove> XML tag reference, 7-54
 <groupLeft> XML tag reference, 7-55
 <labelAttribute> XML tag reference, 7-58
 <layout> XML tag reference, 7-59
 <link> XML tag reference, 7-63
 <matrix> XML tag reference, 7-65
 <matrixCell> XML tag reference, 7-67
 <matrixCol> XML tag reference, 7-68
 <matrixRow> XML tag reference, 7-69
 <object> XML tag reference, 7-71
 <programUnits> XML tag reference, 7-72
 <properties> XML tag reference, 7-74
 <property> XML tag reference, 7-77
 <report> XML tag reference, 7-78
 <section> XML tag reference, 7-81
 <select> XML tag reference, 7-82
 <summary> XML tag reference, 7-87
 <tabular> XML tag reference, 7-88
full URL syntax, 4-8
key mapping, 4-5, 4-6
RWCLI60 command line request, 4-1
simplified URL syntax, 4-8
XML report definitions
 additional objects, 7-11
 formatting, 7-7
 formatting exception, 7-9
 full, 7-12
 hyperlink, 7-10
 PL/SQL, 7-10

F

files
 RDF, 4-7
 REP, 4-7
 RWRUN60 executable, 4-1
 uiprint.txt, 3-15

I

installing

- Oracle9i Application Server, 2-1
- starting as a non-service in Windows NT, 3-3
- starting Oracle Reports Services server on
UNIX, 3-2, 3-13, 3-14, 3-15

K

key map file

- cgicmd.dat CGI, 4-5
- enabling, 4-5
- example, 4-5, 4-6
- mapping entries, 4-6
- mapping URL parameters, 4-6
- using, 4-4
- when to use, 4-4

keywords

- CUSTOMIZE, 7-29
- REPORT, 7-28

L

load balancing

- resubmitting jobs, 6-7
- running reports, 6-7

M

maintenance calendar, A-3

MAXENGINE parameter, 6-2, 6-3

MAXIDLE parameter, 6-5

migration from Web cartridge to CGI, F-1

MINENGINE parameter, 6-2, 6-3

MODE command line argument, 4-3

O

OAS (Oracle Application Server), F-1

Oracle Application Server (OAS), F-1

Oracle HTTP Server powered by Apache, 2-2

Oracle Portal

- component request method, 4-2

Oracle Reports Services

architecture, 1-2

configuring

- for clustering, 6-1
- parameters, C-1
- duplicate job detection, 4-3
- tier, 1-2
- view job status on UNIX, 3-3, 3-15

Oracle Reports Services Queue Manager

- monitoring job status, 3-17
- scheduling jobs to run, 4-9

Oracle Reports Services server

configuring

- UNIX with environment variables, 3-14
- Windows NT with environment
variables, 3-12
- with environment variables, 3-12

creating a service entry, 3-11

database queue, 3-18

setting the default, 3-11, 3-12

UNIX, 3-12

Windows NT, 3-12

starting, 3-1

UNIX, 3-2, 3-13, 3-14

Windows NT, 3-2

Windows NT as a non-service, 3-3

stopping, 3-1, 3-4

UNIX, 3-4

Windows NT, 3-4

Oracle Universal Installer, 2-1

ORACLE_HOME/guicommon6/tk60/ADMIN
directory, 3-15

ORACLE_HOMEREPORT60 directory, 4-5

ORACLE_HOMEREPORT60SERVER

directory, 3-14

ORIENTATION command line argument, 4-3

overviews

clustering, 6-2

XML, 7-2

P

PAGESIZE command line argument, 4-3
parameters
 CACHESIZE, 4-3
 CLUSTERCONFIG, 6-2, 6-5
 ENGLIFE, 6-5
 MAXENGINE, 6-2, 6-3
 MAXIDLE, 6-5
 MINENGINE, 6-2, 6-3
 Oracle Reports Services configuration, C-1
 RWCLI60 command line arguments, B-1
 SOURCEDIR, 3-13, 3-14, 4-7
PARAMFORM command line argument, 4-3
parser error messages for XML, 7-26
performing batch modifications in XML, 7-25
processing
 reports, 1-6
 Web reports, 1-4

Q

queue activity, database, 3-18

R

RDF
 file, 4-7
registry entries, D-1
REP file, 4-7
REPORT
 command line argument, 4-3, 7-3
 keyword, 7-28
report definitions, XML, 7-1
report requests
 building reports, 4-7
 duplicate job detection, 4-3
 methods
 ActiveX, 4-2
 Oracle Portal component, 4-2
 RWCLI60 command line, 4-1
 SRW.RUN_REPORT, 4-2
 URL syntax, 4-2
 running from a browser, 4-8
 specifying request, 4-7
 when servers are clustered, 6-7

report source path setting, 3-13
REPORTS_PATH configuration environment
 variable, 3-14
REPORTS60_CGIMAP environment variable, 4-5
REPORTS60_COOKIE_EXPIRE environment
 variable, 3-15
REPORTS60_DB_AUTH environment
 variable, 3-16
REPORTS60_ENCRYPTION_KEY environment
 variable, 3-16
REPORTS60_PATH environment variable, 3-13,
 3-14, 4-7
REPORTS60_REPORT_SERVER environment
 variable, F-5
REPORTS60_REPORTS_SERVER environment
 variable, 3-12, 3-16, 3-17, 4-6, 4-8
REPORTS60_REPORTS_SERVER environment
 variables, 6-7
REPORTS60_SSLPORT environment variable, 3-16
REPORTS60_SYS_AUTH environment
 variable, 3-16
resubmitting jobs, 6-7
running
 a report request from a Web browser, 3-16
 reports in a clustered configuration, 6-7
 XML
 report definition by itself, 7-25
 report definitions, 7-19
runtime
 customization
 overview, 7-2
 XML report definition, 7-3
rw_server.sql script, 3-18
RWCLI60
 command, 4-1, 4-2
 command line argument, 4-6, 4-8, B-1
 command line request, 4-1
RWRUN60 executable file, 4-1

S

SCHEDULE command line argument, 6-7
script, rw_server.sql, 3-18
security, 5-1

SERVER

- command line argument, 3-16, 3-17, 4-2, 4-6, 4-8, 6-7, F-5
- servlet, 1-5
- setting
 - default Oracle Reports Services server, 3-11
 - UNIX, 3-12
 - Windows NT, 3-12
- SOURCEDIR parameter, 3-13, 3-14, 4-7
- specifying report requests, 4-6
 - by building a report, 4-7
 - by scheduling to run automatically, 4-9
 - from a Web browser, 4-8
- SRW.RUN_REPORT request method, 4-2
- SRW.RUN.REPORT command, 4-2
- starting Oracle Reports Services server, 3-1
 - UNIX, 3-2
 - Windows NT, 3-2
 - Windows NT (non-service), 3-3
- stopping Oracle Reports Services server, 3-1
 - UNIX, 3-4
 - Windows NT, 3-4

T

- tags
 - reference, XML, 7-30
 - XML for report definitions, 7-30
- text conventions, xv
- thin client tier, Oracle Reports Services, 1-2
- THOUSANDS command line argument, 4-3
- TNS_ADMIN
 - configuration environment variable, 3-14
 - environment variable, 3-13
- tolerance, B-23
- TOLERANCE command line argument, 4-3
- tracing options for XML, 7-26

U

- uiprint.txt file, 3-15
- UNIX, Oracle Reports Services server
 - configuring with environment variables, 3-14
 - setting the default, 3-12
 - starting, 3-2
 - stopping, 3-4
- URL syntax
 - adding as a hyperlink, 4-8
 - full syntax example, 4-8
 - hiding command line arguments, 4-4
 - report request method, 4-2
 - running from a browser, 4-8
 - simplified syntax example, 4-8
 - simplifying requests, 4-4
- USERID
 - command line argument, 4-3
 - using, XML report definitions, 7-2

W

- Web
 - CGI, 1-5
 - key map file, 4-5
 - server tier, Oracle Reports Services, 1-2
 - server, configuring, 3-9
- Windows NT, Oracle Reports Services server
 - configuring with environment variables, 3-12
 - setting the default, 3-12
 - starting, 3-2
 - starting as a non-service, 3-3
 - stopping, 3-4

X

XML

- applying
 - multiple report definitions, 7-20
 - report definition at runtime, 7-19
 - report definition in PL/SQL, 7-21
 - report definition stored in a file, 7-21
 - report definition stored in memory, 7-21
- creating
 - report definition, 7-3
 - report definition required tags, 7-4
 - report definitions, 7-2
- debugging report definitions, 7-26
- parser error messages, 7-26
- report definitions, 7-1
 - additional objects, 7-11
 - applying, 7-20
 - applying via PL/SQL, 7-21
 - batch modifications, 7-25
 - debugging, 7-26
 - for building reports, 7-5
 - formatting example, 7-7
 - formatting exception example, 7-9
 - full, 7-12
 - hyperlink example, 7-10
 - overview, 7-2
 - parser, 7-26
 - partial, 7-5
 - PL/SQL example, 7-10
 - required tags, 7-4
 - running, 7-25
 - running to Oracle Reports Services Builder, 7-28
 - tags, 7-30
 - writing to files, 7-29
- running
 - report definition by itself, 7-25
 - report definitions, 7-19
- RWBLD60, 7-28
- tag reference, 7-30
- TEXT_IO, 7-29
- tracing options, 7-26
- using report definitions, 7-2

