Oracle® Application Server Reports Services

Publishing Reports to the Web

10*g* (9.0.4)

Part Number B10314-01

November 2003

ORACLE®

Oracle Application Server Reports Services Publishing Reports to the Web, 10*g* (9.0.4)

Part Number B10314-01

Primary Author: Frank Rovitto

Contributing Authors: Mick Andrew, Robin J. Fisher, Parineetha Henry, Shaun Lin, Avnish Malik, Rohit Marwaha, Vinay Pamadi, Rajesh Ramachandran, Sripathy P Rao, Danny Richardson, Jim Safcik, Anil Sharma, Vishal Sharma, J. Toby Shimizu, Navneet Singh, Jeff Tang, Vanessa Wang, Philipp Weckerle

Contents

Part I   Preparing Your Environment

1   Understanding the OracleAS Reports Services Architecture

2 Starting and Stopping OracleAS Reports Services

3 Configuring OracleAS Reports Services

4   Managing Fonts in Oracle Reports

5 Printing on UNIX with Oracle Reports

6 Using PDF in Oracle Reports

7   Configuring Destinations for OracleAS Reports Services

8   Configuring and Using the JDBC PDS

9   Securing OracleAS Reports Services

10   Configuring and Administering OracleAS Single Sign-On

11   Deploying Reports in OracleAS Portal

12 Clustering Reports Servers

Part II   Sending Requests to the Server

13   Running Report Requests

16 Customizing Reports with XML

17 Using Event-Driven Publishing

Part III National Language Support and Bidirectional Support

18 Implementing NLS and Bidirectional Support

Part IV Performance

19 Managing and Monitoring OracleAS Reports Services

20 Tuning Oracle Reports

Part V   Appendices

A   Command Line Options

B   Reports-Related Environment Variables

C   Batch Registering Reports in OracleAS Portal

Glossary

Index

xx

Send Us Your Comments

Oracle Application Server Reports Services Publishing Reports to the Web, 10*g* (9.0.4)

Part Number B10314-01

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

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

If you find any errors or have suggestions for improvement to this documentation, please send us your comments via the Oracle Reports discussion group forum:

(`http://otn.oracle.com/products/reports/`)

If you have problems with the software, contact your local Oracle Support Services representative.

Preface

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

> **Note:** For the latest updates to this manual, refer to the Oracle Technology Network, (`http://otn.oracle.com/products/reports/`), then click Getting Started and use the index to navigate to *Oracle Application Server Reports Services Publishing Reports to the Web*.

Intended Audience

This manual is intended for anyone who is interested in publishing reports with OracleAS Reports Services. To configure OracleAS Reports Services, it is useful for you to have a solid understanding of the following technologies:

- Your operating system

- Java

- Databases

- CORBA

- JSP files

- XML and DTD files

- Web server configuration

- HTTP

This manual will guide you through configuring components related to these technologies.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation   JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation   This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

This manual contains the following chapters:

Chapter 1, "Understanding the OracleAS Reports Services Architecture"
Provides an overview of OracleAS Reports Services architecture.

Chapter 2, "Starting and Stopping OracleAS Reports Services"
Tells you how to start and stop OracleAS Reports Services.

Chapter 3, "Configuring OracleAS Reports Services"
Describes how to configure the OracleAS Reports Services.

Chapter 4, "Managing Fonts in Oracle Reports"

Describes how to configure, alias, and troubleshoot fonts in Oracle Reports.

Chapter 5, "Printing on UNIX with Oracle Reports"

Describes how to configure for printing and troubleshoot printing problems on Unix.

Chapter 6, "Using PDF in Oracle Reports"

Describes the PDF features in Oracle Reports and how to generate PDF output for multibyte fonts, bidirectional (BiDi) languages, barcode, and Unicode character sets.

Chapter 7, "Configuring Destinations for OracleAS Reports Services"

Explores how OracleAS Reports Services handles output processing to default and custom destinations.

Chapter 8, "Configuring and Using the JDBC PDS"

Provides information on configuring the JDBC PDS for use in OracleAS Reports Services.

Chapter 9, "Securing OracleAS Reports Services"

Provides a conceptual description of OracleAS Reports Services security features.

Chapter 10, "Configuring and Administering OracleAS Single Sign-On"

Describes how to configure and use OracleAS Reports Services with OracleAS Single Sign-On.

Chapter 11, "Deploying Reports in OracleAS Portal"

Describes how to use OracleAS Portal to deploy your reports developed in OracleAS Reports Services.

Chapter 12, "Clustering Reports Servers"

Describes how to cluster Reports Servers to enhance performance and reliability.

Chapter 13, "Running Report Requests"

Describes the various methods for running reports, in particular, how to construct a runtime URL.

Chapter 14, "Using the Oracle Reports Web Service"

Describes the various operations supported by the Oracle Reports Web service.

Chapter 15, "Creating Advanced Distributions"

Describes how set up advanced distributions through a distribution XML file.

Chapter 16, "Customizing Reports with XML"

Provides information about customizing reports at runtime with XML.

Chapter 17, "Using Event-Driven Publishing"

Describes how to use the event-driven publishing to invoke reports automatically in response to database triggers.

Chapter 18, "Implementing NLS and Bidirectional Support"

Provides NLS information for OracleAS Reports Services.

Chapter 19, "Managing and Monitoring OracleAS Reports Services"

Describes OracleAS Reports Services integration with Oracle Enterprise Manager (OEM) and tells you how you can monitor your Reports Servers with OEM.

Chapter 20, "Tuning Oracle Reports"

Provides tips for tuning and performance enhancements.

Appendix A, "Command Line Options"

Provides information about reports-related command line options.

Appendix B, "Reports-Related Environment Variables"

Provides information about Oracle Reports environment variables.

Appendix C, "Batch Registering Reports in OracleAS Portal"

Provides information on registering reports in OracleAS Portal using batch scripts.

Related Documents

For more information on building reports, OracleAS Portal, or OracleAS Reports Services, refer to the following manuals:

- *Oracle Reports Tutorial*, B10612_01

- *Oracle Reports Building Reports*, B10602_01

- *Getting Started with OracleAS Portal*, available on the Oracle Technology Network (`http://portalcenter.oracle.com`)

- Getting Started with Oracle Reports, available on the Oracle Technology Network (`http://otn.oracle.com/products/reports/`)

Conventions

The following conventions are used in this manual:

Convention	Meaning
. . .	Vertical ellipses in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipses in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
variable	Monospace italic type indicates variables or user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Part I

Preparing Your Environment

Part I contains overview information about the OracleAS Reports Services environment and provides practical information about preparing that environment for running reports. This includes starting and stopping OracleAS Reports Services, configuring Reports-related OracleAS Reports Services components, font related information in Oracle Reports, configuring destinations for OracleAS Reports Services, configuring and using the JDBC PDS, setting up OracleAS Reports Services Security, OracleAS Single Sign-On, deploying reports in OracleAS Portal, and Reports Server clusters.

Part I includes the following chapters:

- Chapter 1, "Understanding the OracleAS Reports Services Architecture"

- Chapter 2, "Starting and Stopping OracleAS Reports Services"

- Chapter 3, "Configuring OracleAS Reports Services"

- Chapter 4, "Managing Fonts in Oracle Reports"

- Chapter 5, "Printing on UNIX with Oracle Reports"

- Chapter 6, "Using PDF in Oracle Reports"

- Chapter 7, "Configuring Destinations for OracleAS Reports Services"

- Chapter 8, "Configuring and Using the JDBC PDS"

- Chapter 9, "Securing OracleAS Reports Services"

- Chapter 10, "Configuring and Administering OracleAS Single Sign-On"

- Chapter 11, "Deploying Reports in OracleAS Portal"

- Chapter 12, "Clustering Reports Servers"

1

Understanding the OracleAS Reports Services Architecture

When you're ready to publish your reports, all the Web server and application server tools you'll need are available in the Oracle Application Server. This chapter describes the architecture of relevant OracleAS Reports Services components in combination with its reports publishing component, OracleAS Reports Services. It also provides an overview of Reports Runtime processing and offers some things to consider when you set up your server environment.

This chapter includes the following sections:

- Overview of OracleAS Reports Services
- OracleAS Reports Services Components
- OracleAS Reports Services Runtime Process
- Things to Consider When You Set Up Your System
- Maintaining High Availability

1.1 Overview of OracleAS Reports Services

Oracle Application Server is a comprehensive and integrated application server that runs any Web site, portal, or Internet application. It enables you to make applications available from Web browsers, mobile devices, and command lines. OracleAS Reports Services is the reports publishing component of Oracle Application Server. It is an enterprise reporting service for producing high quality production reports that dynamically retrieve, format, and distribute any data, in any format, anywhere. You can use OracleAS Reports Services to publish in both Web-based and non-Web-based environments.

OracleAS Reports Services provides a scalable, flexible architecture for the distribution and automated management of report generation engines on the same server and across multiple servers. Additionally, it caches report output for reuse on similar requests. It integrates into standard Web environments with JSPs, Java servlets, and CGI. It enables you to run reports on both local and remote application servers and to implement a multi-tiered architecture for running your reports.

When used in conjunction with servlet, JSP, or CGI (maintained only for backward compatibility), OracleAS Reports Services enables you to run reports on any platform from a Web browser using a standard URL syntax. For servlet implementations, the in-process server is available for faster response and easier administration. The in-process server cuts down on the communication expense between processes and consequently increases response times.

OracleAS Reports Services handles client requests to run reports by entering all requests into a job queue. When one of the server's 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 engines until it reaches the maximum limit specified in your server configuration. Similarly, engines are shut down automatically after having been idle for a period of time that you specify (see Chapter 3, "Configuring OracleAS Reports Services").

OracleAS Reports Services keeps track of all jobs submitted to the server, including jobs that are running, scheduled to run, finished, or failed. The Reports Queue Manager (Windows), the Reports Queue Viewer (UNIX), the showjobs command (Web), and the OracleAS Reports Services pages in Oracle Enterprise Manager (OEM) enable you to view information on when jobs are scheduled, queued, started, finished, and failed, as well as the job output and the final status of the report.

With OracleAS Reports Services, job objects are persistent. This means that if the server is shut down then restarted, all jobs are recovered,[1] not just scheduled jobs.

When used in a Web environment, the OracleAS Reports Services architecture consists of four tiers:

Note: The term *tier* refers to the logical location of the components that comprise the OracleAS Reports Services architecture. Each of the tiers, though, could reside on the same machine or different machines.

---

[1] Only synchronous jobs and jobs that are currently running are lost in this case.

- The client tier (a Web browser)

- The Web server tier

- The OracleAS Reports Services tier

- The data tier, including databases and all other data sources

When used in a non-Web environment, there are three tiers (a Web server being unnecessary):

- The client tier (a small, proprietary application on each client machine)

- OracleAS Reports Services tier

- The data tier, including databases and pluggable data sources

The way you set up these tiers can range from having all of them on one machine to having each of them on a separate machine. Additionally, you can have multiple Web servers on multiple machines as well as multiple application servers on multiple machines.

If you choose to have your Web server on multiple machines, the Oracle HTTP Server provides a load balancing feature to allow sharing of the Web server load across multiple machines. If you choose to have your application server on multiple machines, OracleAS Reports Services provides peer-level clustering to allow sharing of the Reports Server load among multiple machines.

The difference between load balancing and peer clustering is that with load balancing, one machine manages the traffic across all machines; while with peer clustering, all machines are aware of the traffic on each machine, and each machine shares the task of monitoring and responding to requests. The advantage of peer-level clustering is the elimination of a single point of failure, further supporting the possibility of a fail-safe environment.

> **Note:** Reports Server clustering is discussed in detail in Chapter 12, "Clustering Reports Servers".

OracleAS Reports Services provides event-based reporting. This uses database events to trigger the generation of a report. For example, you can define an event that signals a change in revenue levels above or below a particular watermark. If the change occurs in the database (the event), a report is automatically generated. This feature is discussed in detail in Chapter 17, "Using Event-Driven Publishing".

OracleAS Reports Services includes a distribution module that uses XML to define unique configurations for the distribution of reports. Call the desired XML file from the runtime command line or URL to generate one report, and let the server handle diverse outputs and destinations. Processing time is significantly reduced and configuration changes can all be handled within the XML file. This feature is discussed in detail in Chapter 15, "Creating Advanced Distributions".

1.2 OracleAS Reports Services Components

Figure 1–1   OracleAS Reports Services Components
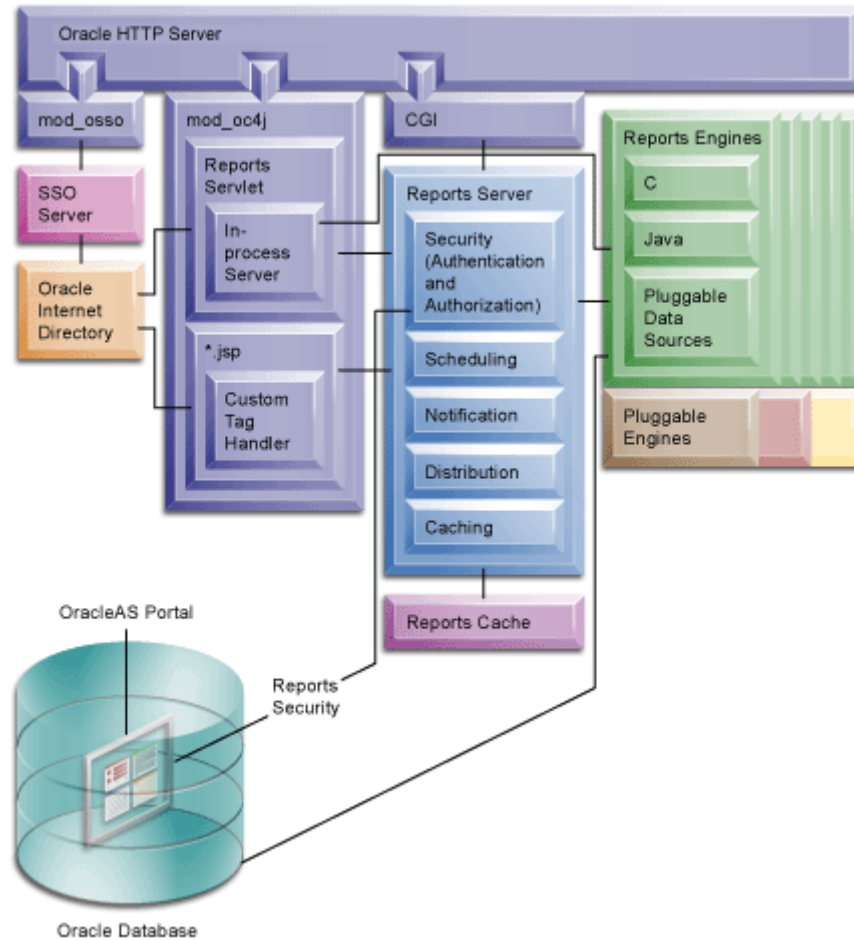
Figure 1–1 illustrates the components of a working OracleAS Reports Services environment. This includes:

1. The **Oracle HTTP Server**, a Web server provided by Oracle Application Server. It incorporates an OpenSSL module to provide support for Secure Sockets Layer

(SSL) and HTTP Secure Sockets Layer (HTTPS). It also provides a servlet engine to support the running of Java Servlet applications.

2. The module **mod_oc4j**, used by the Oracle HTTP Server to redirect requests from servlets and JSPs to Oracle Application Server Containers for J2EE (OC4J). OC4J provides a complete J2EE environment that includes a JSP translator, a JSP servlet engine (OJSP), and an Enterprise JavaBeans (EJB) container. It provides a fast, lightweight, highly scalable, easy-to-use, complete J2EE environment. It is written entirely in Java and executes on the standard Java Development Kit (JDK) Virtual Machine (JVM).

3. The **Reports Servlet,** a component of OracleAS Reports Services that runs inside the Web server's servlet engine. The Reports Servlet translates and delivers information between HTTP and Reports Server. It includes:

   - The **In-Process Server**, which reduces the maintenance and administration of the Reports Server by providing a means for starting the server automatically, whenever it receives the first request from the client through the Reports Servlet (`rwservlet`) or a Reports JSP.

   - The **Custom Tag Handler**, which processes custom Oracle Reports tags included in a JSP file. In a JSP file, Oracle Reports-related custom tags are identified by the prefix `rw:`; other custom tags using other prefixes may also be present.

4. The **Reports CGI**, a component of the Web server that translates and delivers information between either a Web server or a J2EE Container (for example, OC4J) and the Reports Server, enabling you to run a report dynamically from your Web browser .

   > **Note:** Reports CGI is maintained only for backward compatibility.

5. The **Reports Server**, which processes client requests, including ushering them through its various services, such as authentication and authorization checking, scheduling, caching, and distribution (including distribution to custom—or pluggable—output destinations). The Reports Server also spawns runtime engines for generating requested reports, fetches completed reports from the Reports Server cache, and notifies the client that the job is ready.

6. The **Reports Server Cache**, which securely stores completed job outputs.

7. The **Reports Engine**, which includes components for running SQL-based and pluggable data source-based reports. It fetches requested data from the data

source, formats the report, sends the output to cache, and notifies the Reports Server that the job is complete.

8. The **Pluggable Engines**, which are custom engines that use Java APIs to pass jobs to the Reports Server, as well as leverage the server's other features, such as scheduling, distribution, notification, and caching. OracleAS Reports Services provides an out-of-the-box pluggable engine called the URL engine. The URL engine enables you to distribute content from any publicly available URL to such destinations such as e-mail, OracleAS Portal, and WebDav.

1.3 OracleAS Reports Services Runtime Process

Here is how the various components of OracleAS Reports Services contribute to the process of running a report:

1. The client requests a report by contacting a server through either a URL (Web) or a non-Web, Oracle Reports-related command, such as `rwclient`.

   ■ The URL goes to JSP, `rwservlet`, or CGI, all associated with the Oracle HTTP Server. The JSP and `rwservlet` requests go to mod_oc4j. (For jobs run as JSPs, mod_oc4j uses OJSP to translate the JSP into a servlet.) The CGI requests go to a CGI component.

   The URL may contain runtime parameters or a keyword that refers to a runtime parameter configuration section within `cgicmd.dat`, or it may contain both, though parameters explicitly named in the URL must not also be present in the relevant keyword section of `cgicmd.dat`.

   ■ `rwclient` goes directly to the Reports Server.

   The command line may contain runtime parameters. If you have a lot of runtime parameters, you can create a batch file or shell script that contains the `rwclient` command along with a string of parameters.

2. The `rwservlet` or the Reports CGI (`rwcgi`, maintained only for backward compatibility) component translates and delivers information between either a Web server or a J2EE Container (for example, OC4J) and the Reports Server:

   ■ Server requests from JSP or using `rwservlet` can by run by the in-process Reports Server or as a stand-alone Reports Server process, whichever is specified in the servlet configuration file, `rwservlet.properties` (*ORACLE_HOME*\reports\conf\). An in-process server requires less maintenance than a stand-alone server because, unlike the stand-alone server, it starts automatically in response to requests from the client.

Additionally, an in-process server cuts down on the communication between processes, increasing the potential for faster performance.

- Server requests using `rwcgi` go to the stand-alone server.

3. The Reports Server processes the request:

If the request includes a TOLERANCE argument, then the Reports Server checks its cache to determine whether it already has output that satisfies the request. If it finds acceptable output in its cache, then it immediately returns that output rather than rerunning the report.

> **Note:** For any job request that you send to the Reports Server, you can include a TOLERANCE argument. TOLERANCE defines the oldest output that the requester would consider acceptable. For example, if the requester specified five minutes as the TOLERANCE, the Reports Server would check its cache for the last duplicate report output that had been generated within the last five minutes. An expiration argument defines the point in time when the report output should be deleted from the cache (for example, expiration might equal a specific date and time for when the output should expire). For more information, see Appendix A, "Command Line Options".

If the request is the same as a currently running job, then the request will reuse the output from the current job rather than rerunning the report.

If neither of these conditions is met, then the Reports Server processes the request:

a. If configured, the Reports Server initiates an authentication and authorization check through mod_osso, part of the Oracle HTTP Server.

b. If the report is scheduled, the Reports Server stores the request in the scheduled job queue, and the report is run according to schedule. If the report is not scheduled, it is queued in the current job queue for execution when a Reports Engine becomes available.

> **Note:** When you configure the Reports Server (in *server_name*.conf), you can specify the maximum number of the Report Engines it can use. If the Reports Server is under this maximum, then it can send the job to an idle engine or start a new engine to handle the request. Otherwise, the request must wait until one of the current Oracle Reports Engines completes its current job.

    c. At runtime, the Reports Server spawns a Reports Engine and sends the request to that engine to be run.

4. The Reports Engine retrieves and formats the data.

5. The Reports Engine populates the Reports Server cache.

6. The Reports Engine notifies the Reports Server that the report is ready.

7. The Reports Server accesses the cache and sends the report to output according to the runtime parameters specified in either the URL, the command line, or the keyword section in the cgicmd.dat file (URL requests only).

Another way to create a report is through event-driven publishing. With event-driven publishing, the client is the database (rather than the end user). Events are defined through the Event-Driven Publishing API. The event invokes a database trigger, an advanced queuing application, or a PL/SQL package that calls the Event-Driven Publishing API to submit jobs to the Reports Server. Event-Driven Publishing is discussed in detail in Chapter 17, "Using Event-Driven Publishing".

1.4 Things to Consider When You Set Up Your System

The way you set up OracleAS Reports Services can vary widely depending upon the requirements of your system. Before you set up OracleAS Reports Services, you must make some decisions based upon your requirements. By making these decisions beforehand, you can greatly simplify the set-up process.

The following subsections discuss some of the decisions involved in:

- Choosing the Types of Requests You Will Service

- Choosing Servlet, JSP, or CGI

- Choosing Single- Or Multiple-Machine Configurations

- Choosing Whether to Cluster Multiple Servers

1.4.1 Choosing the Types of Requests You Will Service

OracleAS Reports Services can be configured to accept both Web and non-Web job requests.

In the Web case, you can run reports by clicking or typing a URL in a Web browser. Depending on the URL, the report output is then served back to you in your browser or sent to a specified destination (for example, a printer). To enable users to launch reports from a browser, you will use either the Reports Servlet, a JSP, or Reports CGI components with your Web server. One or the other of these components must be present on the Web server to enable communications between it and the OracleAS Reports Services and to enable the processing of report requests from Web clients.

> **Note:** For more information, refer to the Choosing Servlet, JSP, or CGI.

In the non-Web case, you can send job requests using the `rwclient` executable, installed on each of your user's machines.

From the perspective of configuration, these are the key differences between enabling Web and non-Web requests:

- Enabling Web requests requires that you choose between the Reports Servlet, a JSP, or Reports CGI (maintained only for backward compatibility) for the server side, but eliminates the need to install any client software beyond a standard Web browser.

- Enabling non-Web requests requires that you install client software on each machine that will be used to run requests. This introduces the need to administer client software on each client machine.

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. OracleAS Reports Services supports the implementation of both Web and non-Web requests in a single deployment environment.

1.4.2 Choosing Servlet, JSP, or CGI

To use OracleAS Reports Services in a Web environment, you must use a servlet, JSP, or CGI implementation. Our recommendation is that you choose servlet or JSP. The CGI implementation in OracleAS Reports Services is maintained only for backward compatibility.

Between servlet and JSP there are additional considerations. A JSP-only implementation means that you can publish a layout that is optimized for Web delivery. The servlet enables you to include paper layouts in your report publishing solution and fully leverage the distribution features of OracleAS Reports Services.

Using the servlet does not imply that you cannot also use JSP files because JSP files can contain both Web and paper layouts. When you run a report stored in a JSP, you specify the servlet in the URL and call the JSP with the command line argument: `report=`*`myreport`*`.jsp`.

For more information on running reports, see Chapter 13, "Running Report Requests".

1.4.3 Choosing Single- Or Multiple-Machine Configurations

You can place OracleAS Reports Services on the same machine as your Web server or on a different machine. Both scenarios have pros and cons.

For example, while it's true that having OracleAS Reports Services and the Web server on the same machine requires more of the machine's memory and disk space, it's also true that such an implementation reduces network traffic. This is because requests traveling between the Web server and the application server do not have to travel across a network, only incoming requests must do so.

If you are using the in-process server (available only with servlet implementations) you can further amplify the performance advantages of a single machine. The in-process server speeds up processing time by allowing for faster and more efficient communication between OracleAS Reports Services components. We recommend that you use the in-process server unless you will not use the Reports Servlet to deploy reports.

On the other hand, if you have a single machine configuration and that machine fails, everything fails.

While there is a greater amount of network traffic when the Web server and the application server are on different machines, you also benefit from the increase in system resources, in the form of additional CPUs, more disk space, and more available memory. Even in a multiple machine configuration, the in-process server will aid performance by speeding communication between OracleAS Reports Services components

Another possibility is placing your Web server and your application server each on multiple machines. This will require additional configuration, but it allows you to implement load balancing on the Web server.

If you will be deploying reports in multiple languages, you'll want to set up multiple Reports Servers—one or more for each language.

1.4.4 Choosing Whether to Cluster Multiple Servers

A cluster is a virtual grouping of servers into a community for the purpose of sharing request processing efficiently across members of the cluster. Unlike in previous versions, clustering in OracleAS Reports Services is peer-level, rather than master/slave. Peer-level clustering means that all members of the cluster take equal responsibility for sharing and processing incoming requests. Incoming requests are sent to the cluster as a whole rather than any one Reports Server in the cluster. Thus, if one member is shut down, the other members carry on managing the request load. There is no single-point-of-failure, where one machine's malfunction brings the whole system down.

Each cluster member machine must be configured in more or less the same way to allow a report to run on each server member in the same way. This means that configuration files should have most of the same settings: a distinction can be drawn between job-related settings and machine-related settings. Job-related settings must be the same from cluster member to cluster member. Job-related settings include settings related to security, data sources, and destination types. Machine-related settings include such attributes as `maxEngine`, `minEngine`, `maxIdle`, `initEngine`, and the like—these can be different from member to member.

Additionally, for cluster members:

- Server-related environment variables should be set to the same values.
- TNS settings should point to the same databases in the same way.

For servers to be members of the same cluster, they must share a cluster name (appended to each server's server name) and have the same public and private keys.

If your machines require different job-related configuration settings, you will not benefit from clustering.

If you must set your servers up for different languages, you'll want to set up multiple clusters: one or more for each language.

1.5 Maintaining High Availability

Oracle Application Server provides a number of high availability features to keep its middle tier running even when particular servers or components fail. To take

advantage of these high availability features, OracleAS Reports Services takes the following actions when its infrastructure dependencies fail:

- **Retrying the OracleAS Portal database connection.** If the connection from the Reports Server to the OracleAS Portal database schema is dropped for some reason, then the Reports Server tries to re-establish the connection before generating an error. First, it retrieves the OracleAS Portal connection string from the repository. With the OracleAS Portal connection string, the Reports Server can attempt to reconnect. If reconnection is successful, you need not restart the Reports Server. OracleAS Reports Services also supports cold failover and RAC on the infrastructure and disaster recovery on the middle tier.

- **Retrying the Oracle Internet Directory connection.** If the Oracle Internet Directory connection becomes stale for some reason, the Reports Servlet and the Reports Server try to re-establish the connection before generating errors. If reconnection is successful, you need not restart the Reports Server.

> **See Also:** For more information about high availability:
>
> - *Oracle Application Server 10g High Availability Guide*

2

Starting and Stopping OracleAS Reports Services

This chapter provides information on starting and stopping OracleAS Reports Services. It includes the following main sections:

- Starting and Stopping the Reports Server
- Verifying that the Oracle HTTP Server Is Running
- Verifying that the Reports Servlet and Server Are Running

> **Note:** The examples in this chapter use *ORACLE_HOME* to denote where the Oracle Application Server is installed. This includes OracleAS Reports Services.

If you plan to run reports on the Web, you must first start the Oracle HTTP Server. You'll find information on doing this in your Oracle Application Server documentation. When you follow any of the procedures in this chapter, we assume you have already started the Oracle HTTP Server.

2.1 Starting and Stopping the Reports Server

The best way to run the Reports Server is through the Oracle Process Manager and Notification Server (OPMN). OPMN provides a centralized mechanism for initializing, maintaining, and shutting down your Oracle HTTP Server, Oracle Application Server Containers for J2EE processes, and OracleAS Reports Services. For more information about configuring the Reports Server through OPMN, refer to Chapter 3, "Configuring OracleAS Reports Services".

> **Note:** Another advantage of configuring your Reports Server through OPMN is that OPMN will automatically restart a Reports Server if it crashes for some reason.

2.1.1 Starting, Stopping, and Restarting Reports Servers from Oracle Enterprise Manager

When the standalone Reports Server is configured through OPMN, as it is by default, you can start, stop, and restart it through Oracle Enterprise Manager.

> **Note:** The standalone Reports Server is automatically configured in OPMN and thus registered with Oracle Enterprise Manager during installation of Oracle Application Server. If you add any Reports Servers after installing Oracle Application Server, you must register the new server(s) manually in the Oracle Enterprise Manager's `targets.xml` file and the Oracle Process Manager and Notification Server's `opmn.xml` file. For more information, see Configuring Reports Server with the Oracle Process Manager and Notification Server and Oracle Enterprise Manager. The in-process Reports Server is not configured in OPMN.

To start, stop, or restart a Reports Server:

1. In the Oracle Enterprise Manager Application Server Control, go to the **Targets > *host name* > Application Server > Reports Server** for the Reports Server you want to manage.

2. On the Reports Server's main page:

   - Click **Start** to start the server.

   - Click **Stop** to stop the server.

   - Click **Restart** to restart the server.

   These buttons appear on a Reports Server's main page according to the server's current state:

   - When the server is down, the **Start** and **Stop** buttons display.

   - When the server is up, the **Restart** and **Stop** buttons display.

2.1.2 Starting, Stopping, and Restarting Reports Servers from the Oracle Process Manager and Notification Server

You can use the following command lines to start, stop, and restart the Reports Server if it was configured through the Oracle Process Manager and Notification Server:

```
ORACLE_HOME/opmn/bin/opmnctl startproc ias-component=reports_server_name
ORACLE_HOME/opmn/bin/opmnctl stopproc ias-component=reports_server_name
ORACLE_HOME/opmn/bin/opmnctl restartproc ias-component=reports_server_name
```

The Reports Server name must match the name in the ias-component id in the opmn.xml file. For more information about configuring the Reports Server through the Oracle Process Manager and Notification Server, refer to Chapter 3, "Configuring OracleAS Reports Services".

2.1.3 Alternative Methods of Starting and Stopping the Reports Server

If you choose not to run your Reports Server through OPMN and maintain it via Oracle Enterprise Manager, you can use these older methods of running the Reports Server:

- Installing and Starting the Reports Server as a Service (Windows)

- Starting the In-Process Server (Windows and UNIX)

- Starting the Reports Server from a Command Line (Windows and UNIX)

- Stopping the Reports Server

The following subsections tell you how to set up each of these options.

2.1.3.1 Installing and Starting the Reports Server as a Service (Windows)

By default, the Reports Server is installed as an in-process server, but, if you wish, you can install the Reports Server as a service on a Windows machine. To do so, at the command prompt enter:

```
rwserver -install server_name [batch=yes/no] [autostart=yes/no]
```

For batch, the default is no. Enter yes if you do not want to be prompted for confirmation during installation. For autostart, the default is no. Enter yes if you want the service to start automatically at reboot without requiring a user to manually start the Reports Server.

Add the cluster name to this command if this server will be a member of a cluster. For example:

```
rwserver -install server_name.cluster_name [batch=yes/no] [autostart=yes/no]
```

To learn more about clustering servers together, see Chapter 12, "Clustering Reports Servers".

> **Note:** To remove the Reports Server service, type the following at a command prompt: rwserver -uninstall *server_name*. Include the cluster name if the server is a member of a cluster, for example: rwserver -uninstall *server_name.cluster_name*.

To start your Reports Server on Windows:

1. On the machine that hosts the Reports Server, choose **Start > Settings > Control Panel** and double-click **Services** in the Control Panel folder.

2. In the Services dialog box, choose **Oracle *ORACLE_HOME* Reports [repserver]**, where *ORACLE_HOME* is the home directory of the Reports Server and *repserver* is the name of the Reports Server instance.

3. Click **Startup**. The Services dialog window displays.

4. In the Services dialog window, select **This Account** in the **Log On As** section, and select an operating system user name and password. This specifies the user account under which the server process is run.

> **Note:** If you want to output to PostScript or to a printer, then ensure the user running the Reports Server service has access to a default printer. Do this by using a specific, real user who has printer access when you set up the **Log On As** section of your Windows service. Typically, the System Account does not have access to printers.
>
> For that matter, the user running the Reports Server service must have access to anything the server may need. For example, the server may need write access to another drive.

5. Set the **Startup Type** of the service to **Automatic** when the system is started.

6. Click **OK**.

7. Click **Start**.

   A **Service Control** message box indicates when your Reports Server has started.

2.1.3.2 Starting the In-Process Server (Windows and UNIX)

If you are using the Reports Server as an in-process server (the default configuration), sending a run report request starts the in-process Reports Server; however, if you are sending a request via a command line, the servlet must be invoked first using either the run report URL or the Web command URL. When you have successfully started the servlet, this also means you have successfully started the in-process Reports Server.

To directly start the in-process Reports Server from a URL, enter the following from your Web browser:

```
http://your_machine_name:your_port_num/reports/rwservlet/startserver
```

2.1.3.3 Starting the Reports Server from a Command Line (Windows and UNIX)

You can also start the Reports Server as a stand-alone server on Windows using the following command:

```
rwserver server=server_name
```

Add the BATCH command line keyword to start up the server without displaying dialog boxes or messages.

```
rwserver server=server_name batch=yes
```

You can run this command on UNIX using the following syntax:

```
rwserver.sh server=server_name
```

Or:

```
rwserver.sh server=server_name batch=yes
```

You can run this command from any directory as long as the shell script can be reached in your PATH environment variable.

2.1.3.4 Stopping the Reports Server

This section discusses how to stop the Reports Server on Windows and UNIX.

- If the Reports Server is running on Windows as a service, stop it through the Services control panel.

- If the Reports Server running on Windows through the `rwserver` executable, or on UNIX through a shell script, `rwserver.sh`, click **Shutdown** in the Reports Server dialog box.

- Launch Oracle Enterprise Manager, and navigate to the Reports Server you wish to shut down; click **Stop** on the selected Reports Server's home page. For more information about Reports Server and Oracle Enterprise Manager, see Chapter 19, "Managing and Monitoring OracleAS Reports Services".

- If the Reports Server is running as an in-process server through the Reports Servlet, issue the following URL:

  ```
  http://your_host_name:port_number/reports/rwservlet/stopserver
  ```

- If the Reports Server running from a command line on Windows or UNIX, at the command prompt enter the following command:

  For Windows and UNIX (on UNIX use `rwserver.sh` in lieu of `rwserver`):

  The following command line shuts down the server normally (i.e., finishes pending jobs and then stops):

  ```
  rwserver server=server shutdown=normal authid=username/password
  ```

  The following command line shuts down the server immediately (i.e., stops without finishing pending jobs):

  ```
  rwserver server=server shutdown=immediate authid=username/password
  ```

  The following command line shuts down the server without displaying any related messages:

  ```
  rwserver server=server shutdown=normal authid=username/password batch=yes
  ```

  The keywords used with the `rwserver` command are described in Appendix A, "Command Line Options".

> **Note:** `authid` is the Reports Server's administration user name and password. For a secure Reports Server, this user must be a member of the RW_ADMINISTER privilege group in the Oracle Internet Directory. For a non-secure Reports Server, this user is defined in the `identifier` element. The following bullet contains more information on how to stop a non-secure Reports Server using the command line.

- When you stop or shut down a non-secure Reports Server from the command line using either `rwserver.sh` or `rwrqv.sh`, you need to provide a valid `authid`, which must match the value set in the `identifier` element in the server configuration file. However, the `identifier` element is set during Reports configuration while installing Oracle Application Server 10*g* and encrypted by the Reports Server. You can reset the `identifier` element to any value. Then, you also need to change the corresponding properties in `targets.xml` for Oracle Enterprise Manager integration to work. Perform the following steps:

  1. In the non-secure Reports Server's configuration file, *server_name*.conf, modify the `identifier` element to specify the *username/password* and set the `encrypted` attribute to `no`. For example:

     ```
     <identifier confidential="yes" encrypted="no">scott/tiger</identifier>
     ```

  2. Stop and restart the Reports Server manually for the changes made to the *server_name*.conf file to take effect.

     > **Note:** You *must* restart the Reports Server for any configuration changes to take effect.

     The Reports Server will now encrypt the *username/password* value of the `identifier` element. After the Reports Server reads the changes made in the *server_name*.conf file, the following commands should execute successfully (with `scott/tiger` as the *username/password*):

     ```
     ./rwserver.sh server=server_name shutdown=normal authid=scott/tiger
     ./rwrqv.sh server=server_name shutdown=normal authid=scott/tiger
     ```

  3. For Oracle Enterprise Manager integration, edit the `targets.xml` file (in *ORACLE_HOME*/sysman/emd/) using any text editor, as follows:

* Search for target with `TYPE="oracle_repserv"` and `DISPLAY_NAME="Reports Server: server_name"`.

* In the entry, set the `UserName` property and the `Password` property to the same user name and password as in the `identifier` element in the `server_name.conf` file. Set the `ENCRYPTED` attribute to `FALSE` for these two properties.

* Restart Oracle Enterprise Manager for the changes to take effect.

You should now be able to stop and shut down a non-secure Reports Server using Oracle Enterprise Manager.

Note: These steps are required only for a non-secure Reports Server and not for secure Reports Servers..

2.2 Verifying that the Oracle HTTP Server Is Running

You can verify that your Oracle HTTP Server is running from Oracle Enterprise Manager. For more information, refer to your Oracle Enterprise Manager documentation.

Alternatively, you can verify that the Oracle HTTP Server is running, in your browser, by navigating to the following URL:

`http://server_name.domain:port_number/`

2.3 Verifying that the Reports Servlet and Server Are Running

To verify that the Reports Servlet is running, navigate to the following URL:

`http://your_machine_name.domain_name:your_port_number/reports/rwservlet/help`

Note that the URL is case sensitive. If this URL executes successfully, you should get a help page describing the rwservlet command line arguments.

To verify that the Reports Server is running, navigate to the following URL:

`http://your_machine_name.domain_name:your_port_number/reports/rwservlet/showjobs?server=server_name`

The `server=server_name` argument is not required if you are using the default Reports Server name (`rep_machine_name`) or the Reports Server specified in the servlet configuration file, `rwservlet.properties` (`ORACLE_`

HOME\reports\conf\). If this URL executes successfully, you should see a listing of the job queue for the specified Reports Server.

> **Note:** You'll find more information about the servlet configuration file in Chapter 3, "Configuring OracleAS Reports Services".

3

Configuring OracleAS Reports Services

When you install Oracle Application Server, OracleAS Reports Services is configured automatically for you. There will likely be adjustments you wish to make to customize your environment, but you will not be required to set up the entire environment, or even most of it.

This chapter is included largely for reference, should you wish to introduce customizations or have a better understanding of the default configuration. It lists services-related configuration files and describes in detail the content of most of them. It includes the following main sections:

- OracleAS Reports Services Configuration Files

- Configuring Reports Server

- Configuring the Reports Servlet

- Setting the default printer for an in-process server

- Configuring the URL Engine

- Entering Proxy Information

- Configuring Reports Server with the Oracle Process Manager and Notification Server and Oracle Enterprise Manager

- DISPLAY and Printer Dependencies on UNIX

Note: The examples in this chapter use *ORACLE_HOME* to denote where the Oracle Application Server (Oracle Application Server) is installed. Oracle Application Server includes OracleAS Reports Services.

Another aspect of configuration is the setting of environment variables. These are set for you automatically during installation. For reference, environment variables are discussed in Appendix B, "Reports-Related Environment Variables".

3.1 OracleAS Reports Services Configuration Files

This section identifies the various configuration files associated with OracleAS Reports Services. In most cases, you can leave these files untouched. Because they control many aspects of your server environment, you could put that environment at risk if you change a file in some unsupported way. Always keep a back-up of the current version of any configuration file you plan to change.

The configuration files associated with OracleAS Reports Services relate to the Reports Server and the Reports Servlet. They are listed and described in Table 3–1:

> **Note:** The paths specified in Table 3–1 are the same for both Windows and UNIX environments, though they are expressed here using the Windows backslash convention (\).

Table 3–1 OracleAS Reports Services Configuration Files

Component	Configuration File
Reports Server	*ORACLE_HOME*\reports\conf*server_name*.conf
	Use this XML file to define initial values for the Reports Server Cache, the Reports Engine, and security; to register valid destination types; to specify the information to be logged; and to set other server-related values.
	This file is automatically created when you start up the server. If you want to rename your server and wish to keep custom configuration settings you've entered into this file, you must first rename this file to the new server name, then rename the server. Otherwise, the server will create its own new default configuration file.
	You'll find more information about this file in the section Configuring Reports Server.

Table 3–1   OracleAS Reports Services Configuration Files

Component	Configuration File
Reports Server	*ORACLE_HOME*\reports\dtd\rwserverconf.dtd
	This file contains data type definitions for *server_name*.conf and rwbuilder.conf elements and attributes. Data type definitions lists all elements allowed in an associated XML file, the attributes associated with those elements, and default values for those attributes.
	You'll find more information about this file in Reports Server Configuration Elements (rwserverconf.dtd).
Reports Server Reports Builder	*ORACLE_HOME*\reports\conf\rwbuilder.conf
	Use this XML file to configure the Reports Server that is embedded in Reports Builder and Reports Runtime. All run requests must go through Reports Server, meaning that Reports Builder requires a server to run reports. Reports Builder automatically starts a Reports Server to handle its requests. When you run a report from the Builder, this file provides the configuration for the in-process server instance that is invoked. Like the *server_name*.conf file, this file relies on the rwserverconf.dtd file for its data type definitions, though several elements do not apply, including the compatible, persistFile, and security elements.
	Because this file shares most configuration elements found in *server_name*.conf, you'll find the information you need for configuring this file in Configuring Reports Server.
Reports Servlet	*ORACLE_HOME*\reports\conf\rwservlet.properties
	Among other things, this file is where you specify the location and filename of the Reports Servlet key map file (cgicmd.dat) and specify whether you will use the Reports Servlet's in-process server.
	You'll find more information about this file in Configuring the Reports Servlet.

3.2 Configuring Reports Server

The Reports Server component of OracleAS Reports Services is configurable via the XML files *server_name*.conf and rwbuilder.conf, located in the following directory (on both Windows and UNIX):

ORACLE_HOME\reports\conf*server_name* or rwbuilder.conf

Both files are supported by the rwserver.template file, which contains default server configuration values on both Windows and UNIX.

> **Note:** The rwserver.template file is located in the following directory, *ORACLE_HOME*\reports\conf\.

The *server_name*.conf file is the default server configuration file. The rwbuilder.conf file configures the server instance used in-process by Reports Builder.

The *server_name*.conf and rwbuilder.conf files are nearly identical. The only difference between them is that rwbuilder.conf does not use the compatible, persistFile, or security configuration elements, described later in this section, and *server_name*.conf does.

Both of these files are created automatically, under the following circumstances:

- The *server_name*.conf file is created the first time you start Reports Server. It is based on the rwserver.template file.

- The rwbuilder.conf file is created the first time you run a report through Reports Builder. It also is based on the rwserver.template file.

- After you rename the server, a new *server_name*.conf file is created the next time you start the server. The new configuration file is based on the default values present in the rwserver.template file. If you wish to retain the configuration associated with the old server name, you must rename your *server_name*.conf file to the new server name (*new_server_name*.conf), before starting the renamed server.

- If you delete one of these files, the deleted file is recreated the next time you start the server. The new file is based on the default values present in the rwserver.template file.

To explain the syntax and values allowed in these files we'll look at the rwserverconf.dtd file, located in the following directory (on both Windows and UNIX):

ORACLE_HOME\reports\dtd\rwserverconf.dtd

3.2.1 Reports Server Configuration Elements (rwserverconf.dtd)

The following example of rwserverconf.dtd illustrates how it is used to configure various aspects of the Reports Server.

```
<!--
Copyright 2003 Oracle Corp.
```

```
500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A.  All rights reserved.

This is the DTD defining the Reports Server Version 9i Configuration file
(XML) format/syntax.
-->

<!ELEMENT server (compatible?,
                  cache?,
                  engine+,
                  security*,
                  destination*,
                  job+,
                  notification*,
                  log?,
                  jobStatusRepository?,
                  trace?,
                  connection?,
                  queue?,
                  persistFile?,
                  identifier?,
                  environment*,
                  pluginParam*)>
<!ATTLIST server
  version         CDATA         #IMPLIED>


<!ELEMENT cache (property*)>
<!-- class specifies fully qualified java class name which implements
     oracle.reports.cache.Cache interface -->
<!ATTLIST cache
  class           CDATA         "oracle.reports.cache.RWCache">

<!ELEMENT engine (property*)>
<!-- class specifies full qualified java class name which starts engine -->
<!ATTLIST engine
  id              ID            #REQUIRED
  class           CDATA         #REQUIRED
  classPath       CDATA         #IMPLIED
  initEngine      CDATA         "1"
  maxEngine       CDATA         "1"
  minEngine       CDATA         "0"
  engLife         CDATA         "50"
  maxIdle         CDATA         "30"
  callbackTimeOut CDATA         "60000"
  jvmOptions      CDATA         #IMPLIED
```

```
                    engineResponseTimeOut CDATA    "0"
                    defaultEnvId    CDATA          #IMPLIED>


      <!ELEMENT security (property*)>
      <!-- class specifies full qualified java class name which implements
           oracle.reports.server.Security interface -->
      <!ATTLIST security
        id              ID              #REQUIRED
        class           CDATA           #REQUIRED>


      <!ELEMENT destination (property*)>
      <!-- class specifies full qualified java class name which subclass
           oracle.reports.server.Destination abstract class -->
      <!ATTLIST destination
        destype         ID              #REQUIRED
        class           CDATA           #REQUIRED>


      <!ELEMENT job EMPTY>
      <!ATTLIST job
        jobType         CDATA           "report"
        engineId        IDREF           #REQUIRED
        securityId      IDREF           #IMPLIED>


      <!ELEMENT notification (property*)>
      <!ATTLIST notification
        id              CDATA           "mailNotify"
        class           CDATA           #REQUIRED>


      <!ELEMENT log EMPTY>
      <!ATTLIST log
        option          (allJobs|succeededJobs|failedJobs|noJob) "noJob">


      <!ELEMENT jobStatusRepository (property*)>
      <!-- class specifies full qualified java class name which implements
           oracle.reports.server.JobRepository interface -->
      <!ATTLIST jobStatusRepository
        class           CDATA           "oracle.reports.server.JobRepositoryDB">


      <!ELEMENT queue EMPTY>
      <!ATTLIST queue
        maxQueueSize        CDATA           "10000">


      <!ELEMENT connection (orbClient*, cluster?)>
      <!ATTLIST connection
        maxConnect      CDATA           "20"
```

```
      idleTimeOut    CDATA         "10">

<!ELEMENT orbClient EMPTY>
<!ATTLIST orbClient
  id             ID            #REQUIRED
  publicKeyFile  CDATA         #REQUIRED>

<!ELEMENT cluster EMPTY>
<!ATTLIST cluster
  publicKeyFile  CDATA         #REQUIRED
  privateKeyFile CDATA         #REQUIRED>

<!ELEMENT persistFile EMPTY>
<!ATTLIST persistFile
  fileName       CDATA         #IMPLIED>

<!ELEMENT trace EMPTY>
<!ATTLIST trace
  traceFile      CDATA         #IMPLIED
  traceOpts      (trace_prf|trace_brk|trace_app|trace_pls|trace_sql|
                 trace_tms|trace_dst|trace_log|trace_err|trace_inf|
                 trace_dbg|trace_wrn|trace_sta|trace_exc|trace_all) "trace_
all"
  traceMode      (trace_replace|trace_append) "trace_replace"
  traceModule    (all|server|engine) "all">

<!ELEMENT compatible EMPTY>
<!ATTLIST compatible
  version        (6i)          "6i">

<!ELEMENT identifier (#PCDATA)>
<!ATTLIST identifier
  confidential   (yes|no)      "yes"
  encrypted      (yes|no)      "no">

<!ELEMENT environment (envVariable*)>
<!ATTLIST environment
  id             ID            #REQUIRED>

<!ELEMENT envVariable EMPTY>
<!ATTLIST envVariable
  name           CDATA         #REQUIRED
  value          CDATA         #IMPLIED>

<!ELEMENT pluginParam (#PCDATA)>
```

```
<!ATTLIST pluginParam
  name            ID              #REQUIRED
  type            (text|file|url) "text">

<!ELEMENT property EMPTY>
<!ATTLIST property
  name            CDATA      #REQUIRED
  value           CDATA      #REQUIRED
  confidential    (yes|no)   "no"
  encrypted       (yes|no)   "no">
```

The `rwserverconf.dtd` file provides the following elements for configuring the Reports Server:

- server
- compatible
- cache
- engine
- security
- destination
- job
- notification
- log
- jobStatusRepository
- trace
- connection
- queue
- persistFile
- identifier
- pluginParam
- environment

Note that these are XML elements, and XML is case sensitive.

Additionally, when you add any of these elements to the server configuration file (*server_name*.conf), you will save yourself potential error messages from any

XML editor if you use the order of the elements shown in the `rwserverconf.dtd` (*ORACLE_HOME*\reports\dtd\). The configuration file will work regardless of the order, but it will not work if you fail to follow the case specified in `rwserverconf.dtd`.

These elements along with their related attributes and sub-elements are discussed in the following subsections.

3.2.1.1 server

Example
```
<server>
    [One or more configuration specifications]
</server>
```

Required/Optional
Required. You can have a maximum of one open and close `server` element in a given configuration file.

Description
The `server` element opens and closes the content area of the server configuration file. In terms of the file's hierarchy, all the other elements are subordinate to the `server` element.

3.2.1.2 compatible

Example
```
<compatible version="6i"/>
```

Required/Optional
Optional. You can have a maximum of one `compatible` element in your server configuration file.

Description
The `compatible` element is available for backward compatibility with Reports 6*i* clients (RWCLI60, RWCGI60, RWQMU60.EXE, RWRQM60.EXE, RWRQV60.EXE, 6*i* Forms). When `compatible` is set to 6i, Reports Server will make use of an executable file, named `rwproxy`, that listens for requests from a 6*i* client and forwards them to a 9*i* server.

`Compatible` has one attribute: `version`, described in Table 3–2.

Table 3–2   Attributes of the compatible element

Attribute	Valid values	Description
version	6i	Setting `version` to 6i enables Reports 6*i* clients to run under Oracle Reports. When `version` is set to `6i`, versions earlier than 6*i* may also run under Oracle Reports, but they are not certified to do so and are not supported by Oracle.

If you use the `compatible` element, you must also have an entry for Reports Server in the `tnsnames.ora` file as you would have had for the 6*i* version of Reports Server. The installer configures the tnsnames.ora file for the default Reports Server, i.e., `rep_machine_name`.

> **Note:**   The `tnsnames.ora` file is located in the following directory on your Oracle Application Server:
>
> `ORACLE_HOME`/network/admin

For example:

```
testsvr.world = (ADDRESS=
    (PROTOCOL=tcp)
    (HOST=testhost.mydomain.com)
    (PORT=1950)
    )
```

You can bypass this requirement by turning compatibility off. To turn compatibility off, remove the `compatible` element from the Reports Server configuration file.

> **Note:**   If `compatible` is set to `6i`, and you have a TNS entry for Reports Server in your `tnsnames.ora` file, you should include the cluster name if the server is a member of a cluster. If you use a cluster name, you should exclude the default domain that was specified in the `sqlnet.ora` file. For example:
>
> ```
> myserver.world (standalone server with default domain world)
> myserver.cluster1 (server part of cluster1)
> myserver.cluster1.world (invalid entry)
> ```

3.2.1.3 cache

Example

```
<cache class="oracle.reports.cache.RWCache">
    <property name="cacheSize" value="50"/>
    <property name="cacheDir" value="D:\orawin\reports\server\cache"/>
</cache>
```

Required/Optional

Optional. You can have a maximum of one `cache` element in your server configuration file. If no cache element is specified, the default is used (`oracle.reports.cache.RWCache`).

Description

The `cache` element is available for specifying the Java class that defines the server's cache implementation. You can use the default cache Java class or develop your own implementation through the OracleAS Reports Services Cache API.

> **Note:** Look for information about the OracleAS Reports Services APIs on the Oracle Technology Network, (`http://otn.oracle.com`).

The `cache` element has one attribute: `class`, described in Table 3–3.

Table 3–3   Attributes of the cache element

Attribute	Valid values	Description
class	*see the Description column*	**Default:** `oracle.reports.cache.RWCache`
		A fully qualified Java class that implements the `oracle.reports.cache.Cache` interface.

You can also enter from zero to multiple properties under the `cache` element. Properties are name/value pairs recognized and understood by the implementation class you register under `cache`. For example, if you use the default cache Java class that is provided with OracleAS Reports Services, your configuration entry might look like this:

```
<cache class="oracle.reports.cache.RWCache">
    <property name="cacheSize" value="50"/>
```

```
      <property name="cacheDir" value="D:\orawin\reports\server\cache"/>
</cache>
```

In the preceding example, `cacheSize` is measured in megabytes, and `cacheDir`, which points to the location of the cache, is specified for a Windows platform. On UNIX, use UNIX standards, for example:

```
<property name="cacheDir" value="$ORACLE_HOME/reports/server/cache"/>
```

The default cache Java class also provides the following properties:

- `maxCacheFileNumber` is the maximum number of files allowed in the cache. For example:

  ```
  <property name="maxCacheFileNumber" value="250"/>
  ```

- `ignoreParameters` lists any report parameters you want to be ignored when Reports Server constructs the cache key. (The cache key is used by Reports Server to determine if an incoming job request matches existing output in the cache.)

  ```
  <property name="ignoreParameters" value="param1,param2"/>
  ```

3.2.1.4 engine

Example
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="1"
maxEngine="5" minEngine="1" engLife="50" maxIdle="15" callbackTimeOut="60000">
    <property name="sourceDir" value="D:\orawin\reports\myReport"/>
    <property name="tempDir" value="D:\orawin\reports\myTemp"/>
</engine>
```

Required/Optional
Required. You must have at least one `engine` element in your configuration file, and you can have more than one.

Description
The `engine` element identifies the fully qualified Java class that starts an engine and provides a number of attributes that set operational controls on the engine. You can use the default engine provided with OracleAS Reports Services (`oracle.reports.engine.EngineImpl`) or develop your own implementation through the OracleAS Reports Services Engine API. As an example of a custom

engine, you may have developed an engine to execute an operating system command should an event occur in your database.

> **Note:** Look for upcoming information about Oracle Reports APIs on the Oracle Technology Network, (`http://otn.oracle.com`)

The `engine` element has several attributes, described in Table 3–4.

Table 3–4 Attributes of the engine element

Attribute	Valid values	Description
id	string	A keyword, unique within a given configuration XML file, that identifies a particular `engine` element. This can be a text string or a number, for example: `id="rwEng"`
class	*see the Description column*	**Default:** `oracle.reports.engine.EngineImpl` A fully qualified Java class that implements two interfaces: `oracle.reports.engine.Engine` and `oracle.reports.engine.EngineInterface`.

classPath	string	The directory path to the Java class specified in the `class` attribute. To specify the directory, use the conventions required by the server platform, for example:
		
		Windows:
		`classPath="%ORACLE_HOME%\myEngine.jar"`
		
		UNIX:
		`classPath="$ORACLE_HOME/myEngine.jar"`

initEngine	number	**Default:** 1 The number of engines you want Reports Server to start at initialization.
maxEngine	number	**Default:** 1 The maximum number of this type of engine that can run on the server.

Table 3–4   Attributes of the engine element

Attribute	Valid values	Description
minEngine	number	**Default:** 0
		The minimum number of this type of engine that is maintained by the server.
engLife	number	**Default:** 50
		The number of jobs the engine can run before the engine is terminated, and, if necessary, a new engine is started. This feature is available to thwart memory leaks.
maxIdle	number	**Default:** 30
		The number of minutes of allowable idle time before the engine is shut down, provided the current number of engines is higher than minEngine.
		For example, if minEngine is 0, maxIdle is 30, and one engine has been running but unused for 30 minutes, that engine will shut down. If, under the same conditions, minEngine is 1, the active engine will not shut down, even if it has been idle for 30 minutes.
callbackTimeOut	number	**Default:** 60000
		The number of milliseconds of allowable waiting time between when the server launches an engine and the engine calls the server back.
		If the machine that hosts the server is very fast, you can reduce this number for faster performance.

Table 3–4   Attributes of the engine element

Attribute	Valid values	Description
defaultEnvId	string	Is an optional attribute. It specifies the default environment with which Reports Server starts an engine. The attribute takes an id that is associated with an environment element in the server configuration file.
		If you specify defaultEnvId, Reports Server starts an engine with the environment variables specified in the referenced environment element plus whatever environment variables that Reports Server is running under.
		If you do not specify defaultEnvId, Reports Server spawns engines with the environment settings in force at startup time.
		For more information refer to Dynamic Environment Switching.
engineResponseTimeOut	number	**Default:** null (no timeout)
		The maximum amount of time (in minutes) for an engine to update the status of the job while running a report in your environment. If it takes longer than this amount of time to update the job status for some reason (e.g., due to the engine hanging or a long blocking SQL query), Reports Server terminates the job.
jvmOptions	string	The Java Virtual Machine (JVM) options to be used by Reports Server when it starts an engine in the JVM. For example, you can use this attribute to specify the starting heap size and maximum heap size for the JVM, additional classpath entries, and so on.
		The Reports Engine running in the server environment ignores the value in the REPORTS_JVM_OPTIONS environment variable. You need to add the value of the environment variable to jvmoptions attribute in order to let the Reports Engine pick up the JVM options.

Table 3–4 Attributes of the engine element

Attribute	Valid values	Description
keepConnection	YES|NO	**Default:** YES
		The keepConnection property is used by the default runtime engine implementation, i.e., oracle.reports.engine.EngineImpl
		YES The default runtime engine retains the existing database connection information
		NO The default runtime engine discards the existing database connection information and reconnects with the userid specified for the job
		The keepConnection property does not affect reports deployed using either rwbuilder or rwrun.
		This property will be migrated if a *server_name*.conf file used in previous versions (e.g., 9.0.2.x) runs in the current environment.

You can also enter from zero to multiple properties under the engine element. The only requirement is that they be name/value pairs recognized by the Java class that implements the Reports Engine. For example, if you use the default engine Java class that is provided with OracleAS Reports Services, your engine configuration entry might look like this:

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="1"
maxEngine="5" minEngine="1" engLife="50" maxIdle="15" callbackTimeOut="60000">
    <property name="sourceDir" value="D:\orawin\reports\myReport"/>
    <property name="tempDir" value="D:\orawin\reports\myTemp"/>
</engine>
```

In this example, sourceDir and tempDir are set up for a Windows environment (UNIX would be sourceDir="*ORACLE_HOME*/reports/myReport" and tempDir="*ORACLE_HOME*/reports/myTemp"). The sourceDir property identifies the default directory you will use for report definition files. It overrides path information specified in the REPORTS_PATH environment variable.

The tempDir property identifies the name and location of the temporary directory OracleAS Reports Services will use for its temporary files. If this value is unspecified for a default engine, OracleAS Reports Services will use the temporary directory specified in the REPORTS_TMP environment variable. If REPORTS_TMP is also not specified, OracleAS Reports Services will use your operating system's default temporary directory.

The `classPath` attribute is not specified because this configuration uses the default engine class.

3.2.1.5 security

Example

```
<security id="rwSec" class="oracle.reports.server.RWSecurity">
    <!--property name="securityUserid" value="portal_db_username/portal_
    password@portal_db_connection" confidential="yes" encrypted="no"/-->
    <property name="oidEntity" value="oidentity_name"/>
</security>
```

> **Note:** In releases prior to 10*g* (9.0.4), the security element was specified differently. In 10*g* (9.0.4) and later releases, the old property specification, securityUserid, is still provided but commented out. You only need to use securityUserid if you want to connect to an OracleAS Portal instance other than the default instance, which is installed with the Oracle Application Server infrastructure.

Required/Optional

Optional. If you do not enter a security element in the configuration file, Reports Server is not secure. You can have from zero to multiple security elements in your configuration file.

Description

The security element identifies the fully qualified Java class that controls server access. You can use the default security class provided with OracleAS Reports Services, which relies on security features available through OracleAS Portal (included with Oracle Application Server), or develop your own implementation through the Reports Server Security API.

> **Note:** Look for information about Oracle Reports APIs on the Oracle Technology Network, (http://otn.oracle.com).

Security attributes are described in Table 3–5.

Table 3–5   Attributes of the security element

Attribute	Valid values	Description
id	string	A keyword, unique within a given configuration XML file, that identifies a particular `security` element. This can be a text string or a number, for example: `id="rwSec"`
class	*see the Description column*	**Default:** `oracle.reports.server.RWSecurity` A fully qualified Java class that implements Reports Server Security Java interface (`oracle.reports.server.Security`). The default relies on security features available through OracleAS Portal (included with Oracle Application Server).

You also have the option of entering multiple properties under the `security` element. The only requirement is that they be name/value pairs recognized by the Java class that implements Reports Server security. For example, if you use the default security Java class that is provided with OracleAS Reports Services, your `security` configuration entry might look like this:

```
<security id="rwSec" class="oracle.reports.server.RWSecurity">
    <!--property name="securityUserid" value="portal_db_username/portal_
    password@portal_db_connection" confidential="yes" encrypted="no"/-->
    <property name="oidEntity" value="oidentity_name"/>
</security>
```

In this example, the `oidEntity` property illustrates the configuration for 10*g* (9.0.4) and later. The value of `oidEntity` is set by the Installer upon installation. Reports Server uses this entity to connect to the Oracle Internet Directory. Components of the Oracle Application Server can all connect to the Oracle Internet Directory, but each component may have different privileges in the directory. Hence, each component needs to identify itself via its own entity name to the Oracle Internet Directory when it connects. The OracleAS Reports Services entity is of the following format:

```
reportsApp_hostname_GUID
```

For example:

```
reportsApp_testhost.mydomain.com_BBEFDCDAC2343600E0340800020C7BBCC
```

The commented out property, `securityUserid`, illustrates the old method of specifying security. `securityUserid` provides the connection information to enable the Reports Server access to OracleAS Portal security features. The property attributes `confidential` and `encrypted` are available for encrypting the information within the property. Once the `confidential="yes"` and `encrypted="no"` attributes are entered, the property value will be encrypted automatically by Reports Server after you restart the server. When you next open the configuration file, the password information will be scrambled, and `encrypted` will be set to `yes`. If you forget the password you entered in the configuration file, you can delete the property and reenter it with new values, making sure to set `encrypted` to `no`.

> **Note:** For `securityUserid` database connection strings, both the thin (`testhost.mydomain.com:1521:iasdb`) and Oracle Call Interface (`scott/tiger@ordb`) JDBC formats are supported.

When setting up security in a clustered environment, each cluster member should use the same security policy to prevent users from experiencing unexpected behavior.

3.2.1.6 destination

Example

```
<destination destype="oraclePortal"
class="oracle.reports.server.DesOraclePortal">
    <property name="portalUserid" value="portal_db_username/portal_
    password@portal_db_connection" confidential="yes" encrypted="no"/>
</destination>
```

Required/Optional

Optional. If you do not enter a `destination` element in the server configuration file, the provided destination classes will be used (printer, mail, file, cache, and OracleAS Portal—which is an exception in that it requires an entry in the server configuration file so that you may specify the userid and password the server will use to log in to the portal). You can have from zero to multiple `destination` elements in your server configuration file.

Description

Use the destination element to register destination types with the server. There is no need, with the exception of OracleAS Portal, to register provided (default) destinations, such as printers, e-mail, files, or cache. You must register the destination types you create through the OracleAS Reports Services Destinations API.

> **Note:** Look for information about Oracle Reports APIs on the Oracle Technology Network, (http://otn.oracle.com).
>
> Configuring destinations is discussed in detail in Chapter 7, "Configuring Destinations for OracleAS Reports Services".

Destination attributes are listed and described in Table 3–6.

Table 3–6   Attributes of the destination element

Attribute	Valid values	Description
destype	string	Identifies the destination type, for example:
		destype="printer"
class	*See the Description column*	A fully qualified Java class that is a subclass of Reports Server Destination Java class (oracle.reports.server.Destination). Allowable values include:
		oracle.reports.server.DesMail
		oracle.reports.server.DesFile
		oracle.reports.server.DesPrinter
		oracle.reports.server.DesOraclePortal

You also have the option of entering multiple properties under the destination element. The only requirement is that they be name/value pairs recognized by the Java class that is a subclass of the Reports Server Destination Java class. For example:

```
<destination destype="oraclePortal"
class="oracle.reports.server.DesOraclePortal">
    <property name="portalUserid" value="portal_db_username/portal_
    password@portal_db_connection" confidential="yes" encrypted="no"/>
</destination>
```

In this example, the property provides connect information to enable Reports Server to access OracleAS Portal. The `confidential` and `encrypted` attributes are included to automatically invoke encryption on the `portalUserid` value the next time Reports Server is started.

> **Note:** For `portalUserid` database connection strings, both the thin (`testhost.mydomain.com:1521:iasdb`) and Oracle Call Interface (`scott/tiger@ordb`) JDBC formats are supported.

Should your destination implementation require additional information, specify the information in the `pluginParam` element.

3.2.1.7 job

Example

```
<job jobType="report" engineId="rwEng" securityId="rwSec"/>
```

Required/Optional

Required. You must have at least one `job` element and can have more than one.

Description

The `job` element works in collaboration with the `engine` and `security` elements. Use `job` to identify a job type and specify which engine and which security implementation should be used with that type of job. For example, you may have developed an engine to execute an operating system command should an event occur in your database. Using OracleAS Reports Services's event-driven publishing API, you identify the event as a specific job type. When the event occurs, the job type information is sent to Reports Server, which looks up the job type under the `job` element in its configuration file, and follows the direction provided in the element's attributes to the engine (and, if applicable, security implementation) specified for that type of job.

Attributes of the `job` element are listed and described in Table 3–7.

Table 3–7 Attributes of the job element

Attribute	Valid values	Description
jobType	string	**Default:** report
		Describes the type of job to be processed by the server. You can enter any type of job, as long as Reports Server has an engine to process it.
engineId	ID reference	References the ID entered for the engine that will process this job type. Available IDs are specified under the engine element in the server configuration file using the id attribute. The id is a unique keyword (that you devise) within a given configuration XML file that identifies a particular engine.
securityId	ID reference	References the ID entered for the security mechanism that will be applied to this job type. Available IDs are specified under the security element in the server configuration file.

3.2.1.8 notification

Example

```
<notification id="tellMe02" class="oracle.reports.server.MailNotify"/>
```

Required/Optional

Optional. If you do not enter a notification element in the configuration file, the notification function is disabled. You can have from zero to multiple notification elements in your configuration file.

Description

Use the notification element to specify a Java class that defines the type of notification that should be sent when a job succeeds or fails. You can use the default notification class, which provides for notification via e-mail, or design your own with the Oracle Reports Notification API.

> **Note:** Look for upcoming information about Oracle Reports APIs on the Oracle Technology Network, (http://otn.oracle.com).

Attributes of the notification element are listed and described in Table 3–8.

Table 3–8   Attributes of the notification element

Attribute	Valid values	Description
id	string	**Default:** mailNotify
		A keyword, unique within a given configuration XML file, that identifies a particular notification element. This can be a text string or a number, for example:
		id="tellMe01"
class	*See the Description column*	**Default:** oracle.reports.server.MailNotify
		A fully qualified Java class that implements the Reports Server Notification Java class oracle.reports.server.Notification.

If you use the default email notification implementation, use the pluginParam element to specify the outgoing SMTP mail server to be used to send the mail. Use the runtime commands notifysuccess and notifyfailure to specify the email address where notification should be sent (for more information, see Appendix A, "Command Line Options"). For example, you can include these commands in your runtime URL:

```
notifysuccess=recipient's e-mail address&notifyfailure=recipient's e-mail
address
```

With the default e-mail implementation, you can specify only one address for each type of notification. You can specify one or both types of notification. You can send notification each to the same address or each to a different addresses.

A notification element in the server configuration file might look like this:

```
<notification id="mailNotify" class="oracle.reports.server.MailNotify">
    <property name="succNoteFile" value="succnote.txt"/>
    <property name=failNoteFile value="failnote.txt"/>
<notification/>
```

With the default notification implementation, it's not necessary to specify a path to the success or failure text files, provided they're in the default location: *ORACLE_HOME*\reports\templates. Otherwise, enter the directory path along with the filenames according to the requirements of the platform that hosts the server.

3.2.1.9 log

Example

```
<log option="allJobs"/>
```

Required/Optional

Optional. You can have a maximum of one `log` element in your server configuration file.

Description

The `log` element is available for backward compatibility. It invokes the generation and population of a reports log file. The log file is automatically generated and stored in the following path (the path is the same for Windows and UNIX):

ORACLE_HOME\reports*server_name*\rwserver.log

The `log` element has one attribute: `option`, described in Table 3–9.

Table 3–9 Attributes of the log element

Attribute	Valid values	Description
option	allJobs	**Default:** noJob
	succeededJobs	Describes the type of jobs to be included in the log, in addition to default server activities that are logged. Choose from:
	failedJobs	
	noJob	allJobs: all jobs will be logged
		succeededJobs: only jobs that ran successfully will be logged
		failedJobs: only jobs that failed will be logged
		noJob: no jobs will be logged

3.2.1.10 jobStatusRepository

Example

```
<jobStatusRepository class="oracle.reports.server.JobRepositoryDB">
    <property name="repositoryConn" value="scott/tiger@orcl" confidential="yes"
    encrypted="no"/>
</jobStatusRepository>
```

Required/Optional

Optional. You can have a maximum of one jobStatusRepository element in your server configuration file.

Description

The jobStatusRepository element specifies the Java class that implements a job status repository. It provides an additional means (over the persistFile element) of storing job status information.

The persistFile is a binary file and, therefore, cannot be used to publish job status information within your application. The jobStatusRepository provides a means of including status information in your application by providing additional ways of storing it.

The default class, oracle.reports.server.JobRepositoryDB, stores information in a database. Use the Oracle Reports APIs to create your own implementation of the Reports Server Job Repository interface (oracle.reports.server.JobRepository) that stores information wherever you wish.

> **Note:** Look for information about OracleAS Reports Services APIs on the Oracle Technology Network, (http://otn.oracle.com).

The jobStatusRepository element has one attribute: class, described in Table 3–10.

Table 3–10   Attributes of the jobStatusRepository element

Attribute	Valid values	Description
class	*see the Description column*	**Default:** oracle.reports.server.JobRepositoryDB
		A fully qualified Java class that implements the Reports Server Job Repository Java class (oracle.reports.server.JobRepository).

The jobStatusRepository element allows for zero or multiple properties for passing options into the repository. The only requirement is that the class you specify in the server configuration file must recognize the name/value pairs you introduce.

The `jobStatusRepository` element might look like this in your server configuration file:

```
<jobStatusRepository class="oracle.reports.server.JobRepositoryDB">
    <property name="repositoryConn" value="scott/tiger@ORCL" confidential="yes"
    encrypted="no"/>
</jobStatusRepository>
```

In this example, the value for the `repositoryConn` property is the login for access to the database that stores the repository. The `confidential` and `encrypted` attributes are used to invoke encryption on the login information once Reports Server is restarted.

> **Note:** For `repositoryConn` database connection strings, both the thin (`testhost.mydomain.com:1521:iasdb`) and Oracle Call Interface (`scott/tiger@ordb`) JDBC formats are supported.

3.2.1.11 trace

Example

```
<trace traceFile="neptune.trc" traceOpts="trace_prf|trace_dbg|trace_wrn"
traceMode="trace_append" traceModule="server"/>
```

Required/Optional

Optional. You can have a maximum of one `trace` element in your server configuration file.

Description

Use the `trace` element to create a file for tracing your report's execution and to specify the objects and activities you want to trace. The `trace` element controls tracing only for the server and the engine.

> **Note:** Tracing for the servlet and JSP are configured in the servlet configuration file, `rwservlet.properties`, discussed in Configuring the Reports Servlet. Tracing for an individual report can be built into the Reports Runtime command line, discussed in Appendix A, "Command Line Options".

`Trace` attributes are listed and described in Table 3–11.

Table 3–11   Attributes of the trace element

Attribute	Valid values	Description
traceFile	*.trc	**Default:** *server_name*.trc
		The filename of the trace file. If no path is specified, the trace file will be in the following directory on both Windows and UNIX:
		ORACLE_HOME/reports/logs/*server_name*/
traceOpts	see Table 3–12	**Default:** trace_all
		This attribute defines the activities that will be traced. You can have one or more traceOpt values. For example:
		<traceOpts="trace_prf|trace_brk">
		Separate values with a vertical bar (|).Valid values are listed and described in Table 3–12.
traceMode	trace_replace	**Default:** trace_replace
	trace_append	Defines whether new trace information will either overwrite the existing trace file (trace_replace), or be added to the end of the trace, leaving existing trace information intact (trace_append).
traceModule	all	By default, tracing tracks both server and engine events. Tracing engine events can cause performance problems in some cases. Use traceModule to track only server events (server), only engine events (engine), or both (all). For example:
	server	
	engine	
		<trace traceOpts="trace_all" traceModule="server"/>
		If traceModule is not specified, the both server and engine traces are turned on.

Table 3–12   Valid values for the traceOpts attribute

Value	Description
trace_prf	Logs server and engine profile
trace_brk	Lists debug breakpoints
trace_app	Logs information on all report objects
trace_pls	Logs information on all PL/SQL objects
trace_sql	Logs information on all SQL

Table 3–12   Valid values for the traceOpts attribute

Value	Description
trace_tms	Enters a timestamp for each entry in the trace file
trace_dst	Lists distribution lists
	Use this value to determine which report section was sent to which destination.
trace_log	Duplicates log information in your trace file
	If you have specified a log element in your server configuration file, in addition to using the trace element, this value will cause information that is sent to the log file to also be sent to the trace file.
trace_err	Lists server error messages
trace_inf	This is a catch-all option that dumps any information not covered by the other options into the trace file
trace_dbg	Logs debug information
trace_wrn	Lists server warning messages
trace_sta	Provides server and engine state information, such as initialize, ready, run, and shut-down
trace_all	Logs all possible server and engine information in the trace file
trace_exc	Lists all exceptions

When you specify multiple trace elements, separate them with vertical bars. For example:

```
traceOpts="trace_prf|trace_dbg|trace_wrn"
```

3.2.1.12 connection

Example

```
<connection maxConnect="50" idleTimeOut="20">
    <orbClient id="RWClient" publicKeyFile="clientpub.key"/>
    <cluster publicKeyFile="%ORACLE_HOME%\reports\server\serverpub.key"
    privateKeyFile="%ORACLE_HOME%\reports\server\serverpri.key"/>
</connection>
```

Required/Optional

Optional. If you do not specify a connection element in your server configuration file, default values will be used (see Table 3–13). You can have a maximum of one connection element in your server configuration file.

Description

The connection element defines the rules of engagement between the server and the clients connected to it.

Connection attributes are listed and described in Table 3–13.

Table 3–13   Attributes of the connection element

Attribute	Valid values	Description
maxConnect	Number	**Default:** 20
		The maximum number of requests that the server can service simultaneously. Requests in excess of the maxConnect value return a Java exception.
idleTimeOut	Number	**Default:** 15
		Allowable amount of time in minutes the connection can be idle.

In addition to its attributes, connection has two sub-elements: orbClient and cluster.

Use orbClient to provide the name of the public key file that the client will use to connect to Reports Server. Reports Server uses the public key to verify the signature sent by the client when it tries to connect to Reports Server. Reports Server only accepts clients whose signature can be verified through this public key. You can have from zero to multiple orbClient sub-elements in your server configuration file.

The orbClient element attributes are listed and described in Table 3–14.

Table 3–14   Attributes of the orbClient element

Attribute	Valid values	Description
id	string	**Default:** RWClient
		Identifies the Reports Client to be served by the public and private key.

Table 3–14   Attributes of the orbClient element

Attribute	Valid values	Description
publicKeyFile	*filename*.key	**Default:** clientpub.key
		Identifies the public key file that the client will use to connect to Reports Server. Reports Server uses the public key to verify the signature sent by the client when it tries to connect to Reports Server. Reports Server only accepts clients whose signature can be verified through this public key. The default file is stored in the rwrun.jar file.

OracleAS Reports Services provides default client public and private key files, clientpub.key and clientpri.key. These key files are in place for all components of OracleAS Reports Services You can regenerate public and private key files to replace the default key pair. To do this, at the command prompt use the following command:

Microsoft Windows

```
rwgenkey.bat path_and_client_public_key_file_name
            path_and_client_private_key_file_name
```

UNIX

```
rwgenkey.sh path_and_client_public_key_file_name path_and_client_private_key_
file_name
```

If you regenerate these keys, you can specify the public key file locations with the publicKeyFile attribute, and replace the private key file in *ORACLE_HOME*\jlib\zrclient.jar. To do this, you must unjar the file, place the regenerated private key into it, and rejar the file.

Use the cluster sub-element to specify the public and private key files to be used for all cluster members. You can have zero or one cluster element in your server configuration file.

> **Note:** For more information on server clusters, see Chapter 12, "Clustering Reports Servers".

For servers to be members of the same cluster, they must share the same extended cluster name and public and private keys. This means that their extended cluster names (e.g., *serverA*.clus, *serverB*.clus—in this case, .clus is the extended

cluster name) should be the same, and the same public and private key files should be specified in each cluster member's server configuration file (*server_ name.*conf).

The default server public and private keys are stored in *ORACLE_ HOME*\reports\jlib\rwrun.jar on both Windows and UNIX. However, there is no need to jar the newly-generated public and private keys that will be used for the cluster. Put them anywhere, and specify the absolute path and filename for them in the server configuration file.

3.2.1.13 queue

Example
```
<queue maxQueueSize="1000"/>
```

Required/Optional
Optional. You can have a maximum of one queue element in your server configuration file. If you have no queue element, the default, 1000, will remain in effect.

Description
Use the queue element to specify the maximum number of jobs that can be held in each of the reports queues. OracleAS Reports Services has three queue components:

- a queue of scheduled jobs

- a queue of jobs in progress

- a queue of completed jobs

The queue element provides the allowable value for each of these components.

This element is applicable only to the completed job queue. Thus, if the number of jobs exceeds the specified maximum value, that completed job queue will automatically purge its oldest jobs. The scheduled job queue and the in-progress job queue remain unaffected.

You can also use Reports Queue Manager to manually reduce the number of jobs held in the queue.

> **Note:** For more information, see the *Reports Queue Manager online help*.

The queue element has one attribute: maxQueueSize, described in Table 3–15.

Table 3–15   Attributes of the queue element

Attribute	Valid values	Description
maxQueueSize	Number	**Default:** 1000 The maximum number of jobs that can be held in a given reports job queue.

3.2.1.14 persistFile

Example

```
<persistFile filename="neptune.dat"/>
```

Required/Optional

Optional. If you do not specify a file, the server will create one of its own with the default name *server_name*.dat. You can have a maximum of one persistFile element.

Description

The persistFile element identifies the file that records all job status. It is used by Reports Server to restore the server to the status it held before shutdown.

It is named persistFile because the file remains intact, or persists, even when the server is brought down and restarted.

The server persistent file is created automatically the first time you start the server or the first time you start the server after the current server persistent file has been deleted or renamed. If you want to rename this file but continue using it, enter the new name in the server configuration file before you actually rename the file, then restart the server.

The persistFile element has one attribute, fileName, described in Table 3–16.

Table 3–16   Attributes of the persistFile element

Attribute	Valid values	Description
fileName	string	**Default:** *server_name*.dat
		The name and, optionally, the path of the server persistent file. You can leave the path off if the file is kept in its default directory:
		ORACLE_HOME\reports\server\
		The path is the same for Windows or UNIX.

3.2.1.15 identifier

Example

```
<identifier confidential="yes"
encrypted="yes">fpoiVNFvnlkjRPortn+sneU88=NnN</identifier>
```

Required/Optional

Optional. You can have a maximum of one identifier element in your server configuration file.

Description

The identifier element is automatically written to the configuration file when you first log in to Reports Queue Manager as an administrator. The first login sets the Reports Queue Manager's administrator user ID and password. That information is encrypted and written to the server configuration file, then used for authentication for all future Queue Manager logins.

If you forget the Queue Manager login or need to change it, delete it and reenter it in Reports Server configuration file in the following format:

```
<identifier confidential="yes" encrypted="no">username/password</identifier>
```

You should immediately start Reports Server after making this change. Doing so automatically encrypts the user name and password and resets encrypted to yes. It will look something like this:

```
<identifier confidential="yes"
encrypted="yes">fpoiVNFvnlkjRPortn+sneU88=NnN</identifier>
```

> **Note:** This user name and password is also used for accessing
> rwservlet Web commands, such as getjobid, getserverinfo,
> showjobs, and showenv when DIAGNOSTIC=NO in the
> rwservlet.properties file. When DIAGNOSTIC=NO, Web
> commands are disabled for everyone except those administrators
> who have this user name and password.

For more information on Reports Queue Manager, see the *Reports Queue Manager
online help*. For more information on rwservlet.properties, refer to
Configuring the Reports Servlet.

3.2.1.16 pluginParam

Example

```
<pluginParam name="mailServer">smtp01.mycorp.com</pluginParam>
```

Required/Optional

Optional. You can have as many pluginParam elements as you require.

Description

The pluginParam element works in cooperation with all pluggable components of
OracleAS Reports Services. This includes the engine, security, cache, destination,
and jobstatusRepository components. Any one of these may need access to a mail
server, an FTP URL, or some other type of plugin parameter. The pluginParam
element provides a means of specifying plugins that can be used by all pluggable
components. This spares you the task of including this information in the class
definition of the pluggable component and allows you to rapidly and easily change
the source of the plugin.

For example, your custom pluggable engine and destination Java classes may both
need access to a proxy server. Instead of hard-coding access to the server in both of
these classes, you can merely call the type of plugin parameter you need, for
example proxy, and point to its location under pluginParam in the server
configuration file.

You can put in any plugin parameter and name it in any way as long as it is a
plugin supported or required by the pluggable component, and the pluggable
component knows its name.

The pluginParam attributes are listed and described in Table 3–17.

Table 3–17   Attributes of the pluginParam element

Attribute	Valid values	Description
name	string	The name of the plug-in parameter.
type	text	**Default:** text
	file	Describes the type of plugin being specified.
	url	For text, put in the string that is required to identify the named plugin parameter, for example, the name of a mail server. Text means the content of the pluginParam element is text, so the getPluginParam() method will return the exact content specified in the element.
		For file, put in the directory path and filename of the plugin parameter file. Use the standards for specifying directory paths appropriate to Reports Server's host machine (either Windows or UNIX). File means that the content of the pluginParam element is a filename, and the getPluginParam() method will return the content read from the file specified.
		For url, put in the full, absolute URL required by the plugin parameter, for example, the full URL to an FTP site. URL means the content of the pluginParam element is a URL, and the getPluginParam() method will return the content read from that URL. The URL you use must reside on the same side of the firewall as OracleAS Reports Services.
		Note that when you have a default type (text), it is not necessary to specify it in the pluginParam string. The example that heads this section doesn't specify a type because the plugin parameter, a mail server name, is the default type, text.

3.2.1.17 environment

Example

```
<environment id="JP">
  <envVariable name="NLS_LANG" value="Japanese_Japan.JA16SJIS"/>
  <envVariable name="NLS_CURRENCY" value="¥"/>
  <envVariable name="DISPLAY" value="MyServer.MyCompany.com:0.0"/>
</environment>
```

Required/Optional

Optional. You can have as many environment elements as you require.

Description

The environment element defines the characteristics (i.e., environment variables) that you want to use to establish a particular runtime environment. You may include as many environment elements as you need (e.g., one for each language/territory you need to support). Inside an environment element, you can add as many envVariable elements as required.

By referencing the environment element's id, you invoke its settings. You can reference an environment element id from:

- The defaultEnvId attribute of the engine element in the Reports Server configuration file, to apply the corresponding environment settings to that engine when it starts up. For more information, refer to engine.

- The command line option, ENVID, of your report's job request, which makes the environment settings only effective for that particular report job request.

The environment attributes are listed and described in Table 3–18.

Table 3–18   Attributes of the environment element

Attribute	Valid values	Description
id	string	The name of the environment.

The environment element has one sub-element, envVariable. Each envVariable is specified as a name–value pair. They can be either standard environment variables or user-defined environment variables.

The envVariable element attributes are listed and described in Table 3–19.

Table 3–19   Attributes of the envVariable element

Attribute	Valid values	Description
name	string	Is the name of the environment you wish to use (e.g., NLS_LANG).
value	string	Is the value you want to assign to the environment variable identified with the name attribute.

3.2.2 Dynamic Environment Switching

In the past, Reports Server could only serve reports that were compatible with the environment in place when Reports Server was started. For example, reports had to be compatible with the value of the NLS_LANG parameter at the time Reports Server was started. This restriction meant that you needed to have one Reports Server running for each language you wanted to process. The new environment switching feature eliminates this restriction by enabling one instance of Reports Server to serve reports with any arbitrary environment settings, including language.

To use this feature, you first need to add an environment element to your Reports Server configuration file to establish a particular runtime environment. Once you have an environment element established, you can switch to its settings in either one of the following ways:

- The defaultEnvId attribute of the engine element in the Reports Server configuration file applies the specified environment element to the engine when it starts up. For more information, refer to engine.

- The command line option, ENVID, of your report's job request, which makes the environment settings only effective for that particular report job request. Refer to Examples.

Below is an example of an environment element:

```
<environment id="JP">
  <envVariable name="NLS_LANG" value="Japanese_Japan.JA16SJIS"/>
  <envVariable name="NLS_CURRENCY" value="¥"/>
  <envVariable name="DISPLAY" value="MyServer.MyCompany.com:0.0"/>
</environment>
```

You could invoke this environment element from an engine element as follows:

```
<engine id="rwEng" initEngine="1" minEngine="0" maxEngine="10" engLife="50"
maxIdle="30" defaultEnvId="JP"/>
```

The value JP identifies the environment element created previously in the Reports Server configuration file. The initial engines will be spawned with the environment settings specified in this environment element. For more information on the environment element, refer to engine.

3.2.2.1 Examples

The following examples illustrate the usage of the dynamic environment switching feature:

Example 1

Suppose that you need to run reports in Japanese from your Reports Server. An environment conducive to running reports in Japanese would include:

- `NLS_LANG = Japanese_Japan.JA16SJIS`

- The currency unit would be set to Yen (¥), the currency of Japan.

- If Reports Server is running on UNIX, then `DISPLAY` would also need to be set.

To begin, you would have to add an environment element to your Reports Server configuration file that looks something like the following:

```
<environment id="JP">
  <envVariable name="NLS_LANG" value="Japanese_Japan.JA16SJIS"/>
  <envVariable name="NLS_CURRENCY" value="¥"/>
  <envVariable name="DISPLAY" value="MyServer.MyCompany.com:0.0"/>
</environment>
```

Once the environment element is in place, you could request a report with Japanese output using the following URL:

```
http://yourWebServer:port/reports/rwservlet?server=yourreportsserver
&report=Japanese.rdf&userid=username/passwd@db&desformat=htmlcss
&destype=cache&envid=jp
```

When the URL is submitted to Reports Server, it detects the optional `ENVID` parameter and matches the specified id (in this case, JP) to the corresponding id in its configuration file. If Reports Server already has an engine running with these characteristics, it will reuse the existing engine to process the job. If not, then it spawns an engine using the current environment plus the three environment variables specified in the JP environment element. If spawning a new engine would cause Reports Server to exceed its `maxEngines` setting, Reports Server shuts down an engine before starting a new one. An engine may be shut down even though it has not exceeded its `engLife` setting.

Once Reports Server has an engine with the correct environment running, the job is processed by that engine and the output is routed to the specified `DESTYPE`.

`envid` is an optional parameter. If you do not pass this parameter with the job, Reports Server processes the request using an engine started with the `defaultEnvId` environment. If `defaultEnvId` is not specified for the engine element in your Reports Server configuration file, then the engine will inherit the settings with which the Reports Server instance was started.

Reports Server may forward the request to another server on the cluster in cases where it cannot handle the request at that time. Even though the job may be routed to another server on the cluster, the target server will reuse or spawn an engine with the required environment to process this job.

Example 2

The following example illustrates how to use this environment switching feature to run an Arabic report on the same Reports Server that was used to run the Japanese report in Example 1.

Add another environment element to the Reports Server configuration file as shown below:

```
<environment id="AR">
 <envVariable name="NLS_LANG" value="Arabic_United Arab Emirates.AR8ISO8859P6"/>
 <envVariable name="NLS_CALENDAR" value="Arabic Hijrah "/>
</environment>
```

The Arabic report has to be submitted to Reports Server with the following command line:

```
http://yourWebServer:port/reports/rwservlet?server=yourreportsserver
&report=arabic.rdf&userid=username/passwd@db&desformat=htmlcss
&destype=cache&envid=ar
```

Since the job is submitted with envid=AR, Reports Server finds or starts an engine with the environment specified by element AR in the Reports Server configuration file. The job is processed by the new engine and the output is distributed to the specified destination.

Example 3

The following example illustrates how the environment switching feature could be used in conjunction with a JSP report, i.e., without the Reports Servlet.

Suppose that you have the following environment elements in the Reports Server configuration file:

```
<environment id="UK">
 <envVariable name="NLS_LANG"  value="AMERICAN_UNITED KINGDOM.WE8ISO8859P1"/>
</environment>

<environment id="US">
 <envVariable name="NLS_LANG" value="AMERICAN_AMERICA.WE8ISO8859P1"/>
</environment>
```

If your JSP report uses a format mask such as the following, it means the currency, grouping, and decimal symbols can change according to the environment:

```
<rw:field id="sal" src="sal" formatMask="L999G999D999"/>
```

To run the report using the UK symbols for currency, grouping, and decimal, you would use the following URL:

```
http://myserver:port/test/myjsp?userid=scott/tiger@orcl&envid=uk
```

Note: You could place `envid=uk` into a key in the `cgicmd.dat` file.

3.2.2.2 Usage Notes

- Although this feature is ideal for handling reports of various languages, its application can be much broader. You could use it in any situation where a report requires a particular environment to execute correctly.

- Reports Server will start one or more engines per environment id as and when it gets requests for specific environments. The total number of engines, however, cannot exceed the `maxEngine` specified for that engine type. It is recommended that you set `maxEngine` to a value greater or equal to the number of environment elements specified in the Reports Server configuration file.

- `defaultEnvId` can also be applied to pluggable engines other than rwEng. Reports Server will spawn the pluggable engine with the specified environment id.

- For engines used by the in-process server, the order of precedence for environment variables from highest to lowest is as follows:

  - `reports.sh` (UNIX only)

    Note: If you have modified your current `reports.sh` file, you should save it and, after installing 10*g* (9.0.4), merge your modifications into the version of `reports.sh` installed with the latest version. The latest `reports.sh` contains some required changes.

  - `Environment` element in the Reports Server configuration file

- In the *ORACLE_HOME*/j2ee/OC4J_BI_
  Forms/config/oc4j.properties file: the oracle.home property
  defines the *ORACLE_HOME* setting and oracle.path defines the PATH
  setting.

- In *ORACLE_HOME*/opmn/conf/opmn.xml, the <environment> element
  under <oc4j instanceName="OC4J_BI_Forms" gid="OC4J_BI_
  Forms">

- The system settings and registry (Windows only)

- For engines used by the standalone server, the order of precedence for
  environment variables from highest to lowest is as follows:

  - reports.sh (UNIX only)

    Note: If you have modified your current reports.sh file, you
    should save it and, after installing 10*g* (9.0.4), merge your
    modifications into the version of reports.sh installed with the
    latest version. The latest reports.sh contains some required
    changes.

  - environment element in the Reports Server configuration file

  - The environment set in the console where you start rwserver.sh

  - The system settings and registry (Windows only)

- If the same environment variable that is set in ENVID is also set in reports.sh
  (*ORACLE_HOME*/bin/), Reports Server obtains the environment variable value
  from reports.sh and not from ENVID.

  For example, say you want to set the REPORTS_PATH environment variable to a
  different engine by using the environment switch feature. However, the
  reports.sh file also has the same REPORTS_PATH environment variable set.
  Reports Server will now use only REPORTS_PATH set by reports.sh and not
  the REPORTS_PATH set in ENVID when you pass any request.

  To work around this issue, you must:

  1. Open reports.sh and comment the environment variable value. For
     example, comment the REPORTS_PATH value set in the reports.sh file.

  2. Open the *server_name*.conf file.

3. Copy the environment variable value in the `reports.sh` file to the *server_name*`.conf` file using the `environment id` tags. For example:

```
<environment id="default">
  <envVariable name=REPORTS_PATH value="$ORACLE_
  HOME/reports/templates:$ORACLE_
  HOME/reports/samples/demo:$ORACLE_HOME/reports/integ:$ORACLE_
  HOME/reports/printers"/>
</environment>

<environment id="testenv">
 <envVariable name="REPORTS_PATH"
 value="/private/file_path:$ORACLE_HOME/reports/templates:$ORACLE_
 HOME/reports/samples/demo:$ORACLE_HOME/reports/integ:$ORACLE_HOME/
 reports/printers"/>
</environment>
```

4. Add the `defaultEnvId` value to the appropriate tag in the *server_name*`.conf` file. For example, add the `defaultEnvId` value to the `engine` tag so that the engine starts with the default `REPORTS_PATH`.

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1"
maxEngine="1" minEngine="0" engLife="50"
maxIdle="30"callbackTimeOut="60000"
defaultEnvId="default">
```

5. Now run the report.

- In previous releases of Oracle Reports on Solaris, you were required to add the following line in the `reports.sh` file:

```
NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1; export NLS_LANG
```

This workaround was introduced because OPMN implicitly sets `NLS_LANG` to `AMERICAN_AMERICA.WE8MSWIN1252`, which is passed to the engine if `NLS_LANG` is not set. This caused a problem in Solaris because the encoding doesn't exist. However, the `NLS_LANG` setting in `reports.sh` breaks the environment switching feature and will result in any of the following errors:

```
REP-3000: Internal error starting Oracle Toolkit
```

or

```
REP-56048: Engine rwEng-0 crashed
```

To work around this issue, you must remove the NLS_LANG entry in reports.sh.

3.2.3 Connecting to OracleAS Portal

By default, Reports Server can only use portal users to connect to OracleAS Portal. It cannot use an ordinary userid, such as scott/tiger, unless you first assign appropriate privileges to its schema.

To assign the appropriate privileges to a schema other than the portal schema, you need to run the following script from SQL*Plus as an OracleAS Portal user:

ORACLE_HOME/portal/admin/plsql/wor/rwgrant.sql

Once the script is loaded, it prompts you to enter the connection string for the new schema (e.g., repapp/repapp@orcl). The script then assigns the appropriate privileges to this new schema. You will then be able to specify this connection string in the Reports Server configuration file to connect to OracleAS Portal.

3.3 Configuring the Reports Servlet

Configure the Reports Servlet with a file named rwservlet.properties, located in the following path (Windows and UNIX use the same path):

ORACLE_HOME\reports\conf\rwservlet.properties

You may notice that the servlet uses components you may have become familiar with if you used to employ a CGI implementation.

Use the Reports Servlet configuration file for:

- Specifying the Location of the Key Map File
- Reloading the Key Map File
- Hiding Web Command Output
- Selecting Login Dialog Boxes
- Setting Up Trace Options for the Reports Servlet and JSPs
- Specifying the character encoding for reports output using rwservlet
- Disallowing HTML code specified in the URL from being executed in a browser
- Specifying the pool size for concurrent connections to rwservlet
- Customizing the Appearance of Server Error Messages

- Specifying an In-Process Server

- Identifying the Default Reports Server

- Pointing to Dynamically Generated Images

- Setting Expiration for Database and System Authentication Cookies

- Setting an Encryption Key for the Database and System Authentication Cookies

- Adding Formatting to Diagnostic/Debugging Output

- Defining the rwservlet Help File

- Specifying the Use of OracleAS Single Sign-On

The entries in this configuration file are not case sensitive.

For Windows, note that the servlet configuration file uses double backslashes (\\) in lieu of single backslashes to specify a directory path. The first slash "escapes" the second, which would otherwise have another meaning in this file. For example, in a Windows-based Reports Servlet file, the path:

```
d:\orawin\reports\conf\filename.ext
```

Becomes:

```
d:\\orawin\\reports\\conf\\filename.ext
```

For UNIX, use that platform's standard for specifying directory paths, for example:

```
orawin/reports/conf/filename.ext
```

3.3.1 Specifying the Location of the Key Map File

Your report runtime command line may include information you do not want to expose to your users. Additionally, it may be comprised of a long string of options that is difficult to remember or makes for an ungainly URL.

You have the option of entering a report's command line options in a key map file (cgicmd.dat), then limiting the exposed runtime command to the name of the particular key section in this file that holds the required options.

The key map file is discussed in Chapter 13, "Running Report Requests". Use the Reports Servlet configuration file to list the location of this file.

For example:

```
KeyMapFile=d:\\orawin\\reports\\conf\\cgicmd.dat
```

This example uses the default filename and location. An entry for the location and filename of the key map file doesn't appear by default in the servlet configuration file because the servlet already knows what to look for and where to look for it. If you use a file with a different name and/or different location, you must include a `KeyMapFile` parameter in your servlet configuration file that includes the directory path and filename.

3.3.2 Reloading the Key Map File

Use the `RELOAD_KEYMAP` parameter to specify whether the key map file (`cgicmd.dat`) should be reloaded each time the servlet receives a request.

For example:

```
RELOAD_KEYMAP=yes
```

This is useful if you frequently make changes to the map file and want the process of loading your changes to be automatic. Runtime performance will be affected according to how long it takes to reload the file.

Typically, this parameter is set to `no` in a production environment and `yes` in a testing environment.

3.3.3 Hiding Web Command Output

You may want to provide an authentication mechanism for an unsecured Reports Server so that only an administrator (based on the `authid`) can run a Web command, like `showenv`, `showjobs`, etc. By setting `DIAGNOSTIC=NO` in the `rwserver.properties` file, you can provide just such an authentication mechanism. This authentication information is also used for administrative tasks, for example, stopping the Reports Server from the command line.

> **Note:** Setting `DIAGNOSTIC=NO` only works for non-secured Reports Server. For secure Reports Server users, Reports Server verifies the user's privileges based on the entries in Oracle Internet Directory.

To disable Web command display for an unsecured Reports Server

1. Start a Reports Server.

2. Set `DIAGNOSTIC=NO` in the `rwservlet.properties` file.

3. Access the Reports Server from `rwservlet`, by using any of the Web commands, for example, `getserverinfo`. You must pass an `authid` here.

For example:

```
http://yourwebserver:portnum/reports/rwservlet/getserverinfo?server=aks+auth
id=scott/tiger
```

Note: The Reports Server will save this `authid` in the following directory: `ORACLE_HOME/reports/conf/server_name.conf` file under the `<identifier>` element. The `<identifier>` element is set in the Reports Server configuration file by the first Web command call if it is not defined in the server configuration file and the values are automatically encrypted when the Reports Server is started. The format to preset the `<identifier>` element in the Reports Server configuration file is:

```
<identifier confidential="yes"
encrypted="no">username/password</identifier>
```

The clear text of username and password will be encrypted by Reports Server once it starts up, and the encrypted atttribute is changed to yes to indicate the content is encrypted.

Refer to `identifier`, for more information on the `<identifier>` element.

Now, any access to the reports server for using the Web commands will have to pass the same `authid`. If the `authid` is not passed, then you will get the following error:

```
REP-52262: Diagnostic output is disabled.
```

To modify the administrator username and / or password:

1. Open the Reports Server configuration file, i.e., `server_name.conf`.

2. Modify the `<identifier>` element:

```
<identifier encrypted="no"confidential="yes">admin_name/admin_
password</identifier>
```

> **Tip:** You can specify the administrator `username/password` in the `<identifier>` element in the `rwserver.template` file. This ensures that you can pass the same authid for any Reports Server that you start up.

3.3.4 Selecting Login Dialog Boxes

The servlet configuration file offers a number of parameters dealing with templates for userid/password dialog boxes that should open when a user logs in to a database or runs a secure report. Generally, these parameters point to various templates to be used for setting up your login screens. You can customize these templates with your company logo, linked buttons, and/or any other HTML you care to use.

The `DBAUTH` and `SYSAUTH` parameters are for specifying the location and filename of the HTML templates to be used for individual login screens. By default, the file names are `rwdbauth.htm` and `rwsysauth.htm`, respectively.

For example, the following entry points to the template for the database login screen:

```
DBAUTH=rwdbauth.htm
```

`SYSAUTH` points to a login screen for a secure report. For example:

```
SYSAUTH=rwsysauth.htm
```

It isn't necessary to enter the path to a template when it is stored in the default template directory:

```
ORACLE_HOME\reports\templates
```

3.3.5 Setting Up Trace Options for the Reports Servlet and JSPs

Trace has been added to the OracleAS Reports Services environment to allow the logging of many different types of runtime information on various OracleAS Reports Services components.

> **Note:** Tracing for Reports Server is configured in the server configuration file, *server_name*.conf, discussed in Section 3.2.1.11. Tracing for an individual report can be built into the Reports Runtime command line, discussed in Command Line Options.

If you wish to track and log runtime information on the Reports Servlet and JSPs, use the TRACEOPTS parameter in the servlet configuration file. You can enter from zero to multiple trace options. Separate each option with a vertical bar.

For example:

```
TRACEOPTS=trace_prf|trace_pls|trace_dbg
```

All available trace options are listed and described in Table 3–12.

Additionally, you can use the TRACEFILE and TRACEMODE parameters.

Use TRACEFILE to specify the filename of the trace file. For example:

```
TRACEFILE=myrwservlet.trc
```

The default name is rwservlet.trc. If no path is specified, the trace file will be in the following directory on both Windows and UNIX:

```
ORACLE_HOME\reports\logs
```

Use TRACEMODE to define whether new trace information will either overwrite the existing trace file (trace_replace), or be added to the end of the trace, leaving existing trace information intact (trace_append). TRACEMODE replaces or appends to the tracing information that has accumulated since the startup of the Oracle Application Server Containers for J2EE container that contains rwservlet. For example:

```
TRACEMODE=trace_append
```

The default for TRACEMODE is trace_replace.

3.3.6 Specifying the character encoding for reports output using rwservlet

From Oracle Reports 10*g* (9.0.4) onwards, you can specify non-ASCII escaped characters in the request URL or in the Parameter Form input. You must specify the character encoding in the rwservlet.properties file before you can apply it.

This is to ensure that `rwservlet` uses the required encoding when parsing the parameter value.

You can set the value of DEFAULTCHARSET in the `rwservlet.properties` file to either:

- The NLS characterset (for example, `JA16EUC`)

- The IANA charset (for example, `EUC-JP`)

Example   `DEFAULTCHARSET=JA16EUC`.

3.3.7 Disallowing HTML code specified in the URL from being executed in a browser

Any HTML code included as part of the report request URL might lead to a security compromise as it causes certain browsers to execute any script or code in the URL. From Oracle Reports 10*g* (9.0.4) onwards, HTML code is not allowed as part of the URL command.

To disallow HTML code as part of the URL command, a new property ALLOWHTMLTAGS is introduced in the `rwservlet.properties` file. This property is set to `NO` by default, disallowing any HTML code to be entered in the URL when running a report.

Valid Values:

`YES` Allows HTML code in the URL

> **Note:**   Setting `ALLOWHTMLTAGS=YES` allows malicious HTML code to be executed by certain browsers.

`NO` Disallows any HTML code in the URL

Default: `NO`

3.3.8 Specifying the pool size for concurrent connections to rwservlet

You may want to define the number of users who can connect and submit job requests simultaneously to `rwservlet`. To do so, you must set the value of the CONNECTION_POOLSIZE property in the `rwservlet.properties` file.

Default Value   `500`

Minimum Value   0 (not recommended)

Set the value of CONNECTION_POOLSIZE keeping in mind the number of active users expected to make concurrent job requests. For example, set this value to 250 if you expect around 100 concurrent active users.

> **Note:**   The value set (for example, 100) is the number of active users simultaneously making job requests and not the number of users connected to the system without submitting requests.

3.3.9 Customizing the Appearance of Server Error Messages

OracleAS Reports Services provides a template for server error messages. These messages are generated automatically, according to cause. The template provides the visual setting within which the error message is displayed.

You may wish to customize the appearance of error messages, for example with your company logo, or with an icon you plan to associate with errors. You may wish to add buttons that link your users to a help system, your company home page, or back to the last browser window. You can do this by providing your own HTML framework for automatically generated error messages.

The entry in the servlet configuration file is for pointing to the name and location of your error message template.

By default, the entry is:

```
ERRORTEMPLATE=rwerror.htm
```

It isn't necessary to enter the path to the error message template when it is stored in the default template directory:

```
ORACLE_HOME\reports\templates
```

3.3.10 Specifying an In-Process Server

If you choose to run the Reports Server within the same process as the Reports Servlet, you indicate that with the SERVER_IN_PROCESS parameter. To run the Reports Server as an in-process server, specify the following in the servlet configuration file:

```
SERVER_IN_PROCESS=yes
```

Enter no if you do not want the Reports Server to run within the same process as the Reports Servlet.

> **Note:** The pros and cons of running an in-process server are explored in Chapter 1, "Understanding the OracleAS Reports Services Architecture".

3.3.11 Identifying the Default Reports Server

The Reports Servlet uses the SERVER parameter to identify the default Reports Server. If a server name is not specified, for example, in the runtime URL, the default server specified here is used. Enter the name of your default Reports Server in the servlet configuration file.

For example:

```
SERVER=server_name
```

If the default Reports Server is a member of a server cluster, use the cluster name:

```
SERVER=server_name.cluster_name
```

If you use a combination of the server name and cluster name in your URL, requests sent to the default server will go to this specific machine. If this machine is down, an error message will be returned and the report will not be run. By specifying just the cluster name in the URL, requests will be sent to a random cluster member, and fowarded to another if the target machine doesn't have an idle engine.

If you don't specify a SERVER parameter in rwservlet.properties, the default server name is rep_machine_name.

3.3.12 Pointing to Dynamically Generated Images

Optionally, you can use the IMAGE_URL parameter to specify where the reports' dynamically generated images can be accessed.

For example:

```
IMAGE_URL=http://server_or_web_server_name.domain_name:port/reports/rwservlet
```

This parameter is in place for JSPs that do not run via the Reports Servlet. It ensures that dynamically generated images, such as charts, will be viewable only by the person who runs the report. JSPs, and other report types, that run through the Reports Servlet have this protection automatically.

3.3.13 Setting Expiration for Database and System Authentication Cookies

Use the COOKIEEXPIRE parameter to set the lifetime (in minutes) of the database and system authentication cookie. For example:

```
COOKIEEXPIRE=20
```

The default is 30.

Cookies save encrypted user names and passwords on the client-side when users first authenticate themselves. When the server receives a cookie from the client, the server compares the time saved in the cookie with the current system time. If the time is longer than the number of minutes defined in COOKIEEXPIRE, the server rejects the cookie and returns to the client the authentication form along with an error message. Users must re-authenticate to run the report.

3.3.14 Setting an Encryption Key for the Database and System Authentication Cookies

Use ENCRYPTIONKEY to specify the encryption key to be used to encrypt the user name and password of the database and system authentication cookies. The encryption key can be any character string. For example:

```
ENCRYPTIONKEY=egbdf
```

3.3.15 Adding Formatting to Diagnostic/Debugging Output

The DIAGBODYTAGS and DIAGHEADTAGS parameters are available for including additional HTML encoding in the <body> and <head> tags in the output files associated with diagnostic and debugging output.

DIAGBODYTAGS defines the entire <body> tag; while DIAGHEADTAGS defines tags to appear between the open and close <head>/</head> tags.

You can use these to include formatting options to make diagnostic and debugging output easier to read. For example:

```
DIAGBODYTAGS=<BODY [additional HTML encoding]>
```

```
DIAGHEADTAGS=<HEAD>[additional HTML encoding]</HEAD>
```

3.3.16 Defining the rwservlet Help File

A `HELP` keyword is available with the rwservlet command for bringing up a servlet-related help topic. The help file is invoked when you specify the following URL:

```
http://your_web_server/your_servlet_path/rwservlet/help
```

> **Note:** For more about the `HELP` keyword, see Appendix A, "Command Line Options".

We provide a default help file for the servlet, *ORACLE_ HOME*\reports\templates\help.htm, which will be displayed if you leave this parameter undefined. You may want to supply a help file of your own. To do this, specify the name and location URL of your servlet help file with the `HELPURL` parameter in the servlet configuration file. For example:

```
HELPURL=http://your_web_server/your_help_file_path/helpfile.htm
```

3.3.17 Specifying the Use of OracleAS Single Sign-On

If you plan to take advantage of OracleAS Reports Services' Single Sign-On capability, you must ensure the `SINGLESIGNON` parameter is set to `YES` in the servlet configuration file. `SINGLESIGNON` is set to `YES` by default upon installation.

For more information, refer to Enabling and Disabling OracleAS Single Sign-On.

3.4 Configuring the URL Engine

Reports Server includes a URL engine that can take the contents of any URL and distribute them. The URL engine allows you to leverage the powerful scheduling and distribution capabilities of Reports Server to distribute content from any publicly available URL to various destinations such as e-mail, OracleAS Portal, and WebDav. Since Reports Server's destinations are pluggable, you can also add your own custom destinations for the URL content.

Furthermore, if you use the URL engine in conjunction with Reports Server's event-based APIs, database events can trigger the content distribution. For example, suppose you have created a JSP report for high fidelity Web publishing of data stored in a table containing employee expense data. You could then use the URL engine and the event-based API to e-mail that JSP whenever the expense application stores new or updated employee expense data in the table.

If the URL engine is not activated, you can activate it by doing the following:

1. Add an engine element for the URL engine to the server configuration file. For example, your engine element might be as follows:

```
<engine id="rwURLEng"
            class="oracle.reports.engine.URLEngineImpl"
            initEngine="1"
            maxEngine="1"
            minEngine="0"
            engLife="50"
            maxIdle="30"
            callbackTimeOut="60000"
/>
```

2. Add a job element that associates the appropriate job types with the URL engine to the server configuration file. For example, your job element might be as follows:

```
<job  jobType="urlEngine"
        engineId="rwURLEng"
/>
```

3. Stop and restart Reports Server.

> **Note:** When you restart your Reports Server with these new elements, you should see the number of engines increase accordingly in the Reports Server status message box. In the above example, the number of engines would increase by one (the value of initEngine) when you restart Reports Server.

To learn about sending requests to the URL engine, refer to Chapter 13, "Running Report Requests".

3.5 Entering Proxy Information

Some features of OracleAS Reports Services support retrieving or sending information through a firewall. For example, the URL engine, the XML data source, the Text data source, and the mail destination features all retrieve or send information through the firewall. For these features to function properly, Reports Server requires certain proxy information. In the interests of simplicity, you store the necessary proxy information in a single location and point to it from the Reports

Server configuration file. To configure your Reports Server with proxy information, you do the following:

1. Add the `pluginParam` element to the server configuration file and have it point to the proxy information file (e.g., `proxyinfo.xml`). For example, your `pluginParam` element might be as follows:

```
<pluginParam name="proxy" type="file">proxyinfo.xml</pluginParam>
```

> **Note:** You can optionally specify a path for the proxy information file. By default, this file is located in *ORACLE_HOME*/reports/conf.

2. Update the proxy information file with the necessary proxy values for your configuration. For example, `proxyinfo.xml` might contain the following:

```
<proxyInfo>
   <proxyServers>
     <proxyServer name="xyz.abc.com" port="80" protocol="http"/>
     <proxyServer name="www-proxy1.xyz.abc.com" port="80" protocol="ftp"/>
     <proxyServer name="www-prox21.xyz.abc.com" port="80" protocol="https"/>
   </proxyServers>
   <bypassProxy>
     <domain>*.abc.com</domain>
   </bypassProxy>
</proxyInfo>
```

> **Note:** Refer to the default proxy information file, *ORACLE_HOME*/reports/conf/proxyinfo.xml, for additional information.

3.6 Configuring Reports Server with the Oracle Process Manager and Notification Server and Oracle Enterprise Manager

The best way to start, shutdown, monitor, and manage Reports Server is through the Oracle Process Manager and Notification Server (OPMN) and Oracle Enterprise Manager. OPMN provides a centralized mechanism for initializing, maintaining, and shutting down your Oracle Application Server components, including Reports Server. Oracle Enterprise Manager, included with Oracle Application Server, provides managing and monitoring services to OracleAS Reports Services. You can conveniently monitor your Reports Servers through Oracle Enterprise Manager

and, if the process crashes for any reason, OPMN restarts Reports Server for you automatically.

During installation of Oracle Application Server, Reports Servers are automatically configured in OPMN and registered with Oracle Enterprise Manager. If you add any Reports Servers after installing Oracle Application Server, you should register the new server(s) in two places:

- The Oracle Process Manager and Notification Server's `opmn.xml` file.

- The Oracle Enterprise Manager's `targets.xml` file.

To register a new Reports Server in both `opmn.xml` and `targets.xml`, run the following command line:

On UNIX:

```
ORACLE_HOME/bin/addNewServerTarget.sh reports_server_name
```

On Windows:

```
ORACLE_HOME\bin\addNewServerTarget.bat reports_server_name
```

You can add a Reports Server to OPMN using `addNewServerTarget.sh` (UNIX) or `addNewServerTarget.bat` (Windows). However, to view the Reports Server on the Enterprise Manager page, you must restart OPMN by using the following commands:

```
opmnctl stopall
opmnctl startall
```

3.6.1 opmn.xml

Components are configured with OPMN in the `opmn.xml` file located in *ORACLE_HOME*/opmn/conf. To configure Reports Server through OPMN, you need the following in `opmn.xml`:

- a process module

- for the in-process Reports Server, a specification of Reports Server within the desired Oracle Application Server Containers for J2EE component

- for the standalone Reports Server, a Reports Server component

> **See Also:** For a detailed description of OPMN configuration and the contents of `opmn.xml`:
>
> - *Oracle Application Server 10g Administrator's Guide*

3.6.1.1 Process Module

The module tag is included by default in `opmn.xml` and tells OPMN that it needs to load a particular module. In the case of Reports Server, the OracleAS Reports Services module must be loaded. This module is loaded with the following information by default in `opmn.xml`:

```
<module path="/private/oraclehome/opmn/lib/libopmnreports">
  <module-id id="ReportsServices"/>
</module>
```

3.6.1.2 Standalone Reports Server Specification

In the case of the standalone Reports Server, the Reports Server is running in its own component. Therefore, you must specify a separate component for Reports Server. For example:

```
<ias-component id="<RSName>" status="enabled" id-matching="false">
  <process-type id="ReportsServer" module-id="ReportsServices">
    <process-set id="<RSName>" restart-on-death="true" numprocs="1">
      <environment>
        <variable id="PATH" value="your_shell_path"/>
      </environment>
      <module-data>
        <category id="general-parameters">
          <data id="batch"  value="yes"/>
        </category>
        <category id="restart-parameters">
          <data id="reverseping-timeout"  value="120"/>
        </category>
      </module-data>
      <dependencies>
        <OID infrastructure="true"/>
        <database infrastructure-key="portal"/>
        <managed-process ias-component="OC4J" process-type="OC4J_BI_Forms"
          process-set="default_island" autostart="true"/>
        <managed-process ias-component="HTTP_Server"
          process-type="HTTP_Server" process-set="HTTP_Server"
          autostart="true"/>
        <managed-process ias-component="WebCache" process type="WebCache"
          process_set="WebCache" autostart="true"/>
      </dependencies>
      <start timeout="120" retry="3"/>
      <stop timeout="120"/>
      <restart timeout="120"/>
      <ping timeout="30" interval="30"/>
```

```
      </process-set>
    </process-type>
</ias-component>
```

> **Note:** The timeout values in the above example are all in number of seconds.

The key segments of this specification for Oracle Reports are described below.

```
<ias-component id="<RSName>" ...>
```

This tag specifies the name of Reports Server. It must match the Reports Server internal name from `targets.xml`.

> **See Also:** Chapter 19, "Managing and Monitoring OracleAS Reports Services"
>
> For more information about `targets.xml`.

```
<process-type id="ReportsServer" module-id="ReportsServices">
```

This tag defines the process for the named Reports Server and associates it with the OracleAS Reports Services process module.

```
<process-set id="<RSName>" restart-on-death="true" numprocs="1">
```

This tag defines the process characteristics for the named Reports Server. It indicates whether Reports Server should be restarted when it crashes. It also specifies the number of Reports Servers started for this process set, which has to be 1 because the process-set id identifies a single Reports Server name.

```
<variable id="PATH" value="your_shell_path"/>
```

The first tag specifies the value for the PATH environment variable for the process. This variable must be set for the start script to find `uname`. This environment element is not needed on the Microsoft Windows platform.

```
<category id="general-parameters">
  <data id="batch"  value="yes"/>
</category>
```

This group of tags gathers together all of the data (parameters) common to the process. In this particular example, it provides a way to specify that the BATCH parameter be sent to Reports Server. `batch=yes|no` is an option to the start and

stop commands of Reports Server. If it is not configured, this option is not passed in to Reports Server.

```
<category id="restart-parameters">
  <data id="reverseping-timeout"  value="120"/>
</category>
```

This group of tags indicates the restart parameters category, which defines parameters to be used in detecting whether the process has failed and needs to be restarted. If a notification is not received within the specified `reverseping-timeout` period, then the process is considered failed and will be restarted.

```
<dependencies>
```

This tag delimits the list of components upon which Reports Server depends. For example, Reports Server typically depends upon, among other things, the Oracle HTTP Server and Oracle Application Server Containers for J2EE.

OPMN uses dependencies to determine whether to start a process. Like `module-data` and `environment` blocks, `dependencies` blocks can be defined for multiple elements within `opmn.xml`. OPMN creates an aggregate dependency list at the process set level that contains all of the dependencies defined at or above it. If duplicate dependencies are defined at different levels, then duplicate checks on that dependency are made before starting a process.

OPMN has two primary types of dependencies: external and internal. External dependencies are resources not managed by OPMN (the database, Oracle Internet Directory, and OracleAS Single Sign-On). For external resources, an external program performs the check on the resource. Internal dependencies are OPMN-managed processes, which may include processes managed by a remote OPMN instance. Internal dependencies are indicated in the list by the `managed-process` tag.

OPMN maintains a cache of dependency states that contains the last known state of each dependency and the time it was last checked. A single cache entry exists for each dependency with identical attributes, even if that dependency is specified in multiple locations (i.e., for different process sets). A cache timeout parameter for each dependency allows users to specify how long to use its state in the cache. Likewise, a general timeout parameter for each dependency determines how long OPMN should wait for a status update before aborting the dependency check and the process start.

Dependencies are checked in the order in which you declare them. The traversal of this list of dependencies concludes either when the full sequence of checks

completes successfully (the resource is available) or when one of the checks fails (the resource is not available or the check timed out).

The following example tags illustrate a typical list of dependencies for Reports Server:

```
<OID infrastructure="true"/>
<database infrastructure-key="portal"/>
<managed-process ias-component="OC4J" process-type="OC4J_BI_Forms"
     process-set="default_island" autostart="true"/>
<managed-process ias-component="HTTP_Server"
     process-type="HTTP_Server" process-set="HTTP_Server"
     autostart="true"/>
<managed-process ias-component="WebCache"
     process-type="WebCache" process-set="WebCache"
     autostart="true"/>
```

The `OID` tag indicates that Reports Server uses the default Oracle Internet Directory instance for this Oracle Application Server installation.

The `database` tag points to the OracleAS Portal instance used by Reports Server.

The first `managed-process` tag specifies the Oracle Application Server Containers for J2EE instance used by Reports Server. The second `managed-process` tag indicates the Oracle HTTP Server instance.

> **See Also:** For more information about `opmn.xml` and its contents:
>
> ■ *Oracle Application Server 10g Administrator's Guide*

3.6.1.3 In-Process Reports Server Specification

In the case of the in-process Reports Server, the Reports Server is running inside the OC4J component. If you are using the in-process Reports Server, then, within the OC4J component, you must specify the Reports Server data. For example:

```
<ias-component id="OC4J">
  <process-type id="OC4J_BI_Forms">
    <module-data>
      <category id="urlping-parameters">
        <data id="/reports/rwservlet/pingserver?start=auto" value="200"/>
      </category>
    </module-data>
    <process-set .../>
  </process-type>
</ias-component>
```

The key segments of this specification are described below.

```
<category id="urlping-parameters">
```

`urlping-parameters` is a category that identifies all of the URLs to be pinged by the OC4J module. The protocol used for pinging is AJPv1.3.

```
<data id="/reports/rwservlet/pingserver?start=auto" value="200"/>
```

`/reports/rwservlet/pingserver?start=auto` is the URL to be pinged by the OC4J module. In the context of the in-process Reports Server, pinging this URL allows OPMN to determine whether the Reports Server application is responsive. If it is unresponsive, OPMN restarts the corresponding OC4J process.

`value=200` specifies a valid HTTP code (200) that is expected in response to the ping request. If the response HTTP code matches the value configured here, OPMN considers the application healthy and responsive. Otherwise, OPMN restarts the OC4J process.

3.7 DISPLAY and Printer Dependencies on UNIX

In previous releases of OracleAS Reports Services on UNIX, you had to set the DISPLAY environment variable for Reports Server. Previous releases required DISPLAY to use the windowing system display surface for creating images and getting pixel resolution. This dependency is removed in OracleAS Reports Services 10*g* (9.0.4).

Additionally, previous releases required a valid printer on UNIX for fonts. When no valid printer was available, OracleAS Reports Services used the screen fonts, which again required setting the DISPLAY environment variable. In Release 10*g* (9.0.4), OracleAS Reports Services includes a default screen printer surface, ScreenPrinter, that emulates a screen or printer for fonts in the absence of an available printer. As a result, OracleAS Reports Services does not require a printer on UNIX as of OracleAS Reports Services 10*g* (9.0.4).

By default, the environment variable REPORTS_DEFAULT_DISPLAY is set to YES, which specifies that OracleAS Reports Services should:

- remove the dependency on the DISPLAY environment variable (UNIX only)

- use ScreenPrinter for surface resolution for images and font information (UNIX only)

- enable the Advanced Imaging Support (all platforms)

If you wish to revert to the old behavior from previous releases, you can set
REPORTS_DEFAULT_DISPLAY=NO.

> **See Also:** Appendix B, "Reports-Related Environment Variables"

For more information on Oracle Reports environment variables.

3.7.1 ScreenPrinter

The PostScript printer driver `screenprinter.ppd` provides surface resolution for
images and specifies font information. This driver is the first entry in
`uiscreenprint.txt`. The file locations (UNIX only) are:

```
uiscreenprint.txt : ORACLE_HOME/guicommon9/tk90/admin
screenprinter.ppd : ORACLE_HOME/guicommon9/tk90/admin/PPD
```

ScreenPrinter is used for:

- Surface resolution when REPORTS_DEFAULT_DISPLAY=YES.
- Removal of the printer dependency.

If, when generating report output, there is no valid printer queue available (not
found from `TK_PRINTER`, `ORACLE_PRINTER`, `PRINTER`, or `uiprint.txt`), the
surface based on `screenprinter.ppd` will be created and used to get font
information. You can modify the `Fonts` section of `screenprinter.ppd` to
include new fonts, and modify the `DefaultResolution` field to change the
resolution (`DefaultResolution` is 96).

> **Note:** If you do add new fonts, ensure that the new AFM files are
> placed in the AFM directory.

The font look up algorithm on UNIX is:

```
if a valid printer available then
  look up font information from the printer
else
 create a screenPrinter surface
 look up font information from ScreenPrinter
 if ScreenPrinter creation fails then
    REP-1800 : Formatter Error if REPORTS_DEFAULT_DISPLAY is set
  else
    use Screen Fonts
```

> **Note:** In certain multi-byte languages like Chinese, you may want to use screen fonts. However, this would necessitate setting the DISPLAY variable for running the report.
>
> To revert to DISPLAY and use screen fonts (old font look up algorithm):
>
> - Set REPORTS_DEFAULT_DISPLAY=NO
> - Remove the screenprinter.ppd entry in the uiscreenprint.txt file.

See Also:

- Managing Fonts in Oracle Reports
- Printing on UNIX with Oracle Reports

For more information about fonts and printing on UNIX.

3.7.2 Advanced Imaging Support

The quality of images contributes considerably to the overall appearance of a report, particularly for a Web report. You may prefer different image formats in your report output depending on the needs of your project. For example, an aeronautical firm might prefer the higher quality of JPEG or PNG images in their Web reports instead of GIF images. On the other hand, if you are building a Web portal, you might prefer GIF images because of their smaller size and faster download. Similarly, you may wish to import images of these various formats into your report.

Depending on the format of your output, you may choose from a variety of formats for your images.

Table 3–20 Image Format Options by Output Type

Report Output	Available Image Format Choices
HTML,HTMLCSS	PNG, JPEG, JPG, GIF
PDF	PNG, JPEG, JPG, GIF
RTF	PNG, JPEG, JPG, BMP

> **Note:** As you choose your image format, you should take into account the quality and size considerations. Typically, the higher the quality of the image format, the greater the size. For example, PNG and JPEG are higher quality than GIF, but they may also require more storage space.

To enable advanced imaging, you must set the REPORTS_DEFAULT_DISPLAY environment variable to YES. The REPORTS_DEFAULT_OUTPUTIMAGEFORMAT environment variable lets you choose the default image type. Users can override the default choice for images with the OUTPUTIMAGEFORMAT command line keyword. For example:

```
rwclient server=my_rep_server report=images.rdf destype=file desformat=html
desname=images.html userid=scott/tiger outputimageformat=PNG
```

Enabling advanced imaging also allows you to import images of these same formats into your report.

> **Note:** Enabling advanced imaging means that you can no longer use the old CGM and OGD formats in outputs of type HTML or HTMLCSS. If you require these formats for input sources, you should set REPORTS_DEFAULT_DISPLAY to NO. This limitation does not apply on the Windows platform.

See Also:

- Appendix B, "Reports-Related Environment Variables"

For more information on environment variables.

- Appendix A, "Command Line Options"

For more information on command line keywords.

3.8 Setting the default printer for an in-process server

The in-process server does not recognize the default printer of a user currently logged on to Windows. This is because the service that runs the in-process server is logged on as the Local System. Therefore, you might face the following issues:

- Printing to default printer fails with the REP_3002 error. For example, the following command:

  ```
  http://myrepsrvr.us.oracle.com:7777/reports/rwservlet?report=myrep.rdf&desty
  pe=printer&desformat=html
  ```

  will result in the following error:

  ```
  Error:"REP-3002: Error initializing printer.  Please make sure a printer is
  installed."
  ```

- Font alignment problems in a PDF file output from an in-process server.

To work around the issue:

1. Open the Windows registry using a registry editor (e.g., regedit.exe).

   > **Note:** Create a backup of the registry before editing it.

2. Navigate to the following key:

   ```
   HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows
   ```

3. Copy the string value of Device for this key. For example:

   ```
   \\MOWGLI\sierra,winspool,Ne02:
   ```

4. Navigate to the following key:

   ```
   HKEY_USERS\.DEFAULT\Software\Microsoft\Windows NT\CurrentVersion\Windows
   ```

5. Paste the Device value copied from HKEY_CURRENT_USER (the string value of Device for this key will be empty).

   > **Note:** This workaround must be applied everytime you alter the value of the Default Printer.

4

Managing Fonts in Oracle Reports

This chapter provides information about fonts in Oracle Reports. In particular, it covers:

- Using Fonts
- Adding Fonts
- Font Configuration Files
- Font Aliasing
- Troubleshooting Font Issues
- Font Types

4.1 Using Fonts

In Oracle Reports, fonts come into play:

- At build time (in Reports Builder)
- At runtime (in the report's output)
- In the user interface of Reports Builder

4.1.1 Fonts in Reports Builder

Reports Builder provides a list of fonts that are available on the system in the font picker box.

Figure 4–1   Font list in Reports Builder

On Windows, the font list is derived from the fonts that are installed on the system along with the fonts available on the current default printer. A small printer icon before the font name identifies printer fonts. True Type fonts are associated with a TTF icon.

On UNIX, the font list is derived by querying the X-server display on which the application is running for the available fonts. The command is similar to the UNIX `xlsfonts` command, which lists all of the available fonts for the X-server display. From this font list, Reports Builder generates a list of usable fonts with the valid style, weight, width, size, and encoding characteristics to match the character set. The character set is driven by the NLS_LANG environment variable. Reports Builder includes only those fonts with an encoding of iso8859-1, unless specified differently in the toolkit resource file, Tk2Motif.MapsCS. For more information on Tk2Motif.MapsCS, refer to Font Configuration Files.

4.1.2 Fonts in Report Output

During report formatting, fonts associated with the layout objects are first checked against the font alias file, uifont.ali, (refer to Font Configuration Files). If an entry in the font alias file is found, the mapped font is used instead of the original one. The mapped font is then looked up in the list of fonts available on the system or printer. If a particular font is not found, Oracle Reports will look for the nearest matching font under the same character set which can be used instead.

4.1.2.1 Font lookup

On Windows, the font lookup mechanism is simple due to the availability of printer drivers, which have the capability of uploading fonts from the system as needed. Any output from Oracle Reports running on Windows will contain fonts from either one of the following:

- The system
- The printer

For this reason, Oracle Reports considers both the printer and the system fonts when looking for the available fonts.

On UNIX, the fonts available for generating output are either one of the following:

- the fonts available on the printer, specifically the fonts defined in the PPD or TFM files

- if no printer is specified, the fonts available in ScreenPrinter, `screenprinter.ppd`.

> **See Also:**
>
> - Font Configuration Files
>
> - ScreenPrinter
>
> For more information about `uiprint.txt` and ScreenPrinter (`screenprinter.ppd`).

4.1.2.1.1 Font lookup algorithm Figure 4–2 illustrates the process of determining the available fonts for generating report output.

Figure 4–2 Font lookup flow
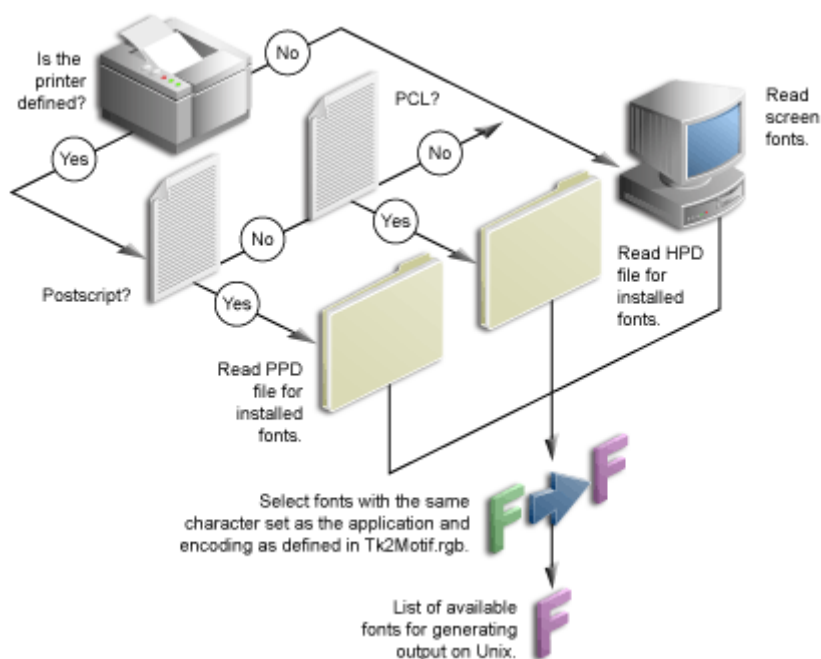
The following steps describe how Oracle Reports generates a list of the available fonts for generating output (e.g., for the screen, printer, or file):

1. Oracle Reports looks in the printer configuration file `uiprint.txt` for all the printers that are listed for the application. If no printers are defined or available, Oracle Reports uses ScreenPrinter.

    See Also:

    - Font Configuration Files

    - ScreenPrinter

    For more information about `uiprint.txt` and ScreenPrinter (`screenprinter.ppd`).

2. Get their type, version and printer definition file.

3. Check for the existence of these printers in the system.

4. If the printers are present, the printer definition files are loaded and the information in these files is read along with the information related to the fonts available for the printer. If these printers are not found, then Oracle Reports uses ScreenPrinter.

5. The AFM files, which are named the same as the font names given in the PPD files, are searched.

6. If found, Oracle Reports then reads this AFM file for all the valid keywords, checks for their correctness and, in case of any discrepancy, default values are used for those keywords.

7. If the AFM file is not found, Oracle Reports marks the font as unusable.

    Note: Similarly for PCL printers, the HP printer definition file (glue file) is loaded and all the fonts defined in the file are also loaded. While looking for a font, Oracle Reports searches for an entry in the HPD file with the font name and accordingly takes the font if the TFM file for this font is also found. Otherwise, Oracle Reports matches this font to the closest available one.

Once the list of available fonts is generated, the mapped font is searched for in this list of fonts and, again, the AFM files are read for the purpose of calculating the text size and weight.

Substituting fonts

If a particular font is needed but not found in the PPD file or if an AFM file is not found, Oracle Reports will look for the nearest matching font according to its matching rules. For example, suppose a report is originally designed with a Korean font SimSun and in the `uifont.ali` file no mapping for this font is found. Oracle Reports will look for the font SimSun in the list of available fonts generated by the Font lookup algorithm. If this font name is not in that list, Oracle Reports tries to look for the closest matching font from the list of fonts given in the printer definition file.

For the SimSun font, the character set is 850. If it cannot find any matching font for this character set, Oracle Reports searches for a font that has a character set for the environment in which the application is running. After a set of fonts with a similar character set is found, Oracle Reports picks the closest match to the requested font based on the font weight, style, etc. If more than one font has the same parameters, Oracle Reports picks the first one and uses it instead of the original font.

Font matching rules

When attempting to match a font, Oracle Reports will try to find the closest match according to the following criteria for fonts with the same character set:

```
fontface > fontsize > fontstyle > fontweight > fontwidth
```

If Oracle Reports can't match the font face, it will try to match the font size; if it can't match the size, it will try to match the font style; and so on.

If a font matches the font size but nothing else and another font matches the style, weight, and width but not the font size, then Oracle Reports will pick the font with the same font size.

It should be noted that irrespective of any font in the output file, the final printed output will depend solely on the fonts installed in the printer.

Example: Suppose that a report has layout objects associated with one of two fonts, Helvetica font of size 8, style Plain, and weight Medium, and Courier font. If the user is running this report on a PostScript-1 printer and generating HTML output, the fonts are chosen as follows:

1. While formatting, Oracle Reports checks `uifont.ali` for any mappings of either specified font. Suppose that `uifont.ali` contains this entry in the [Printer:PostScript1] section:

```
Helvetica.8.Plain.Medium.. = "Mkai-Medium"..
```

Oracle Reports will now search for the Mkai-Medium font instead of the Helvetica font.

2. Oracle Reports looks for the mapped font in the printer definition (PPD) file. Suppose that the PPD file contains the following entry in the *Font Information section:

```
"*Font Mkai-Medium: Standard "(001.004)" Standard ROM"
```

3. Oracle Reports now checks for the associated AFM file (named Mkai-Medium) in the AFM directory. If an AFM file with this name is not found, it will look for another font that has size 8, style Plain, and weight Medium under the same character set as the original font.

Because a report may have to run in many different environments, Oracle Reports always tries to approximate a font for the original font when the original is unavailable. This algorithm is not entirely foolproof. When you create a report, you must be aware of the fonts defined and you should always consider whether those same fonts will be available on the platform where users will run the report. If the font that you have defined is not available in the runtime environment, Oracle Reports substitutes another font that is available on the machine. This process can lead to unexpected and undesirable results, such as strange characters in the report output (e.g., Wingding characters) and incorrect formatting of objects.

If you are encountering these kinds of problems, you should use font aliasing to control the font substitutions made by Oracle Reports. Refer to Font Aliasing.

Oracle Reports follows the above described mechanisms for all output file generation except PDF, which has the PDF font sub setting/embedding capabilities.

> **See Also:** Chapter 6, "Using PDF in Oracle Reports"
>
> For more information on the PDF features and enhancements in OracleAS Reports Services.
>
> For more information on PDF in general, consult the Adobe PDF documentation.

For printing, Oracle Reports generates output based upon the printer driver, in case of Windows, or the printer, in the case of UNIX. On Windows, the output generation is handled by the printer driver. The fonts in this case can either be from the system or from the printer. For fonts which are not available on the printer, the printer will get the fonts from the system through Windows APIs.

4.1.3 Fonts in the User Interface

Text in the user interface of Reports Builder, like the window title, menu items, message boxes, and data model object names, use fonts taken from the system resource files for the current language. These system resource files are supplied with Oracle Reports installation. In Oracle Reports, you can map these fonts in the [RWBUILDER] section of uifont.ali. If found, the mapped font is used instead of the original font. Otherwise, Oracle Reports uses the original font.

On UNIX, these fonts are defined in Tk2Motif.rgb under Tk2Motif*fontList. If the font is not defined, then the default font (fixed for default character set) is used instead. The default system font need not be the one defined in Tk2Motif.rgb. If the defined font does not match the character set on which the application is running, some other available font will be used following the font lookup algorithm discussed in the previous section.

On Windows, in order to maintain the look and feel of the windows, Oracle Reports makes extensive use of the system font, which is obtained from the Windows system parameters. For non-Unicode environments, the font is obtained from the icon objects. You can change it by modifying the fonts via **Display Property > Appearance**. Select Icon from the drop down box and select the desired font name and size. For Japanese Unicode systems, the font is MS Gothic. For Korean, it's MS Sans Serif. For simplified, traditional, and Hong Kong it's Arial. For other languages, it's Lucida Sans Unicode.

You can also change the Windows tool tip font by changing the icon font as described above. This change is not completely reflected across Reports Builder because some tool tip fonts are taken from the resource file.

In Oracle Reports, fonts for the Web Source view are selected by making an entry in the alias file under the [RWBUILDER] section. The entry required for this change should only be aliased to the character set and not to any specific font. For example, if you want to use Arial Unicode MS when NLS is set to UTF8, then you should create an entry like this one:

```
…..UTF8 = "Arial Unicode MS"…..
```

Refer to Font Aliasing for more information.

The supported styles are: Plain, Italic, Oblique, Underline, Outline, Shadow, Inverted, Overstrike, and Blink.

The supported weights are: Ultralight, Extralight, Light, Demilight, Medium, Demibold, Bold, Extrabold, and Ultrabold.

You should not use fonts with a weight of Regular because this weight is not supported and may cause Reports Builder to display undesirable results.

4.2 Adding Fonts

In Oracle Reports, fonts can be added for use:

- At build time (in Reports Builder)
- At runtime (in the output)
- In the user interface
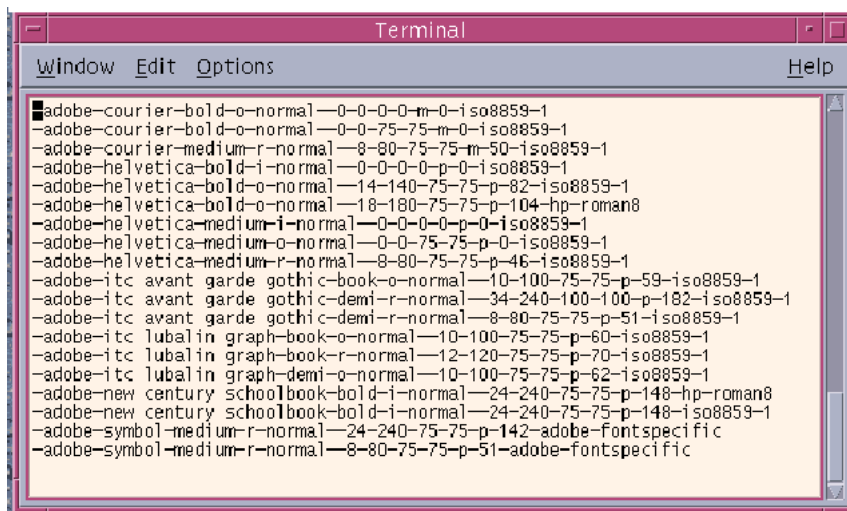
4.2.1 Adding Fonts to Reports Builder

To build a report in a certain font, the font must be available in Reports Builder from the font picker when you are designing the report. In order for the fonts to appear in the font picker, the fonts should be added to the system or the display on which Reports Builder is running. Please review your operating system documentation for adding fonts before attempting this procedure.

To add Type1 fonts on UNIX:

1. Get the font-related files from the vendor. These include the .pfb, .pfa, and the AFM files.

2. Convert the .pfb binary file to .pfa ASCII font using one of the available convertors. Typically, you can get such converters as shareware, e.g., `t1ascii`.

3. Copy the `.pfa` files to the directory where the fonts need to be installed following the instructions for our platform.

4. Verify the installation of the fonts by entering the `xlsfonts -u` command. This command lists all the fonts that are available for that system.

   If you are using a UNIX emulator like reflection X, the fonts installed on the system may not appear in the `xlsfonts` command. The reason for this behavior is that it is taking fonts from the font path or the font server, which is configured for this emulator. If using a font server, make sure that, after installing the font, you add the font directory to the font server configuration file and restart the font server. In the emulator, specify the font path to this font server wherever the fonts are installed. If you are still not able to see the fonts in `xlsfonts`, ensure that the new font directory is the first element of the catalogue in the configuration file.

Figure 4–3   xlsfonts sample output

5. Start Reports Builder on the display that points to the font server on which these fonts are installed or to the display where the fonts are installed.

4.2.2 Adding New Fonts for Report Output

For generating output in Oracle Reports, only the fonts that are specified in the printer definition file are used. To use a newly added font in your output, you should first add it to Reports Builder so that you can assign the font to layout objects when designing the report. Refer to Adding Fonts to Reports Builder for further information.

> **Note:**   If you use fonts in Reports Builder that are not available on your runtime platform, you should alias those fonts on the runtime platform. Refer to Font Aliasing for more information.

The process for adding fonts is different on Windows and UNIX:

- Adding fonts on UNIX
- Adding fonts on Windows

4.2.2.1 Adding fonts on UNIX

To add PostScript fonts:

1. Add the entry for this new font to the *Font Information section in the printer definition (PPD) file:

```
*Font new_font_name Standard '(00.1001)" Standard ROM
*Font ...
```

2. Copy the AFM file for the new font to *ORACLE_ HOME*/guicommon9/tk90/admin/AFM. Ensure that the AFM file name is the same as the *new_font_name* given in the PPD file because Oracle Reports searches for this file based on the font name in the PPD file.

    For example, if the PPD file contains *Font CodedreineunBold: Standard '(00.1001)" Standard ROM, then the AFM file name should be named CodedreineunBold without any extension.

3. If necessary, make changes in the alias file for mapping to this font.

    If the layout objects are associated with the same font name as the new font, then mapping is not required. If the fonts for the layout objects are different and the new fonts are desired in the output file instead of the original ones, then you must map the original fonts to the new ones.

    For example, if the layout objects' font is Helvetica and you want newly installed fonts in the output, then you could add the following to the [Printer:PostScript1] section:

```
Helvetica = CodedreineunBold
```

    Please note the section will be different if you are using a different PostScript level in your uiprint.txt. Refer to Font Aliasing.

To add PCL fonts:

In order to use a new font in Oracle Reports, you need to have the HPD (printer definition files) and TFM files for your printer. The HPD file can be copied from an existing one. You need to be sure that the file is suitable for your printer; fonts referenced in this file should be available on your printer. If the TFM files (fonts) are not available on Oracle Reports installation, you need to contact your font/printer supplier. The new TFM files must be added to the HPD file under a unique font name.

1. In the HPD, you will have to add the new font entry. For example if the new font is Codedreineun then include a new line such as:

```
FONT= Codedreineun
/tfm=9nb17035.tfm
```

2. Copy the associated TFM file into the TFM directory:

```
ORACLE_HOME/guicommon9/tk90/admin/TFM
```

3. Modify the alias file, if necessary, as described in Adding fonts on UNIX for the PostScript printers. The section in which the mapping is done should be [PCL].

4.2.2.2 Adding fonts on Windows

For adding a new font on Windows, refer to your operating system documentation on adding a new font. If the new font has a character set that is compatible with Reports Builder, the new font will appear in the font picker.

4.3 Font Configuration Files

The listing below describes all of the files associated with font configuration for Oracle Reports.

uiprint.txt (UNIX only)

The printer configuration file contains a list of printers installed for the application along with the type of the printer, its version, and the printer definition file. The list of available fonts for runtime is taken from the printer definition file. If no printer is present, Oracle Reports chooses a PostScript printer as the default and default.ppd file as the printer definition file.

> **See Also:**
>
> - Editing uiprint.txt File
>
> - uiprint.txt
>
> For more information on the uiprint.txt file.

Example:
```
Printer: Printer_driver:Driver_specifying_language_and_level:Printer_
description:Printer_definition_file:
```

Each line contains five fields separated by colons.

If you are using PCL printing, then this entry should contain the name of an HPD file.

screenprinter.ppd (UNIX only)

`screenprinter.ppd` is used when a printer is not available on UNIX. `screenprinter.ppd` is in *ORACLE_HOME*/guicommon9/tk90/admin/PPD.

uifont.ali

This file contains mapping information for fonts which can be substituted for other fonts at runtime. Refer to Font Aliasing for more information.

Oracle Reports has added three new sections to the `uifont.ali` file:

`[PDF]` - Used for font aliasing and multi-byte language support

`[PDF:Subset]` - Used for TrueType font subsetting and multi-byte language support

`[PDF:Embed]` - Used for Type1 font embedding

> **Caution:** Do not alter the sections as Oracle Reports parses the `uifont.ali` file looking for keywords. The sections can be in any order.

Some general rules for the `uifont.ali` file are:

- Use double quotes around any font or character set names containing two or more words, or spaces.

- Use # in the first column for comments.

- Comment out lines instead of deleting them to be able to use them in the future.

- Font aliasing is a *font name-to-font name* or *character set-to-CID font* (from Adobe) only.

- Font subsetting is for TrueType fonts only.

- Font subsetting uses the font name and subsets using the TrueType font file name.

- Font embedding is for Type1 fonts only. These fonts have two files. One for metrics, containing either a afm or pfm file extension and the binary containing the pfb file extension.

Font embedding uses the font name and embeds using the Type 1 font file names (both the *FM and PFB files are required in this order).

Refer to Font Aliasing for more information.

PPD and AFM files (UNIX only)

PostScript Printer Definition (PPD) files and AFM (Adobe Font Metrics) files are supplied by Adobe and by printer vendors. These files contain information about the printer. Along with other parameters, these files are read for the information about the available fonts for the printer, which Oracle Reports will use. For all the fonts listed in the PPD file, Oracle Reports searches for the corresponding AFM file according to the font name and loads all of the fonts for which there is an available AFM.

From the fonts perspective, you should modify these files when you add new fonts for the printer and want these changes reflected in Oracle Reports.

Example:

```
*% Font Information =====================
*DefaultFont: Error
*Font AvantGarde-Demi: Standard "(001.001)" Standard
*Font AvantGarde-DemiOblique: Standard "(001.001)" Standard
*Font Courier: Standard "(001.004)" Standard
*Font Courier-Bold: Standard "(001.004)" Standard
```

The AFM files contain information such as the font attributes (style, weight, width, encoding scheme), whether the font is fixed pitch or proportional, and how large each character is.

After looking for the font names from the PPD files, Oracle Reports searches for the AFM files with the same name as the font according to the search criteria described in File Searching. For example, if Oracle Reports finds `AvantGarde-Demi:` `Standard` in the PPD file, it will search for an AFM file named `AvantGarde-Demi` in the AFM directory.

Please note that the AFM files are NOT font files, they are metrics files, which give Oracle Reports information on how to properly format the character for the printer. If you have an AFM file, but the font is not available on the printer, then Oracle Reports cannot generate the font.

Since the AFM files are NOT fonts themselves, if you wish to have more PostScript printer fonts available, you need to:

1. Purchase the fonts and have them installed on the printer.

2. Obtain revised AFM and PPD files from the font/printer vendor.

3. Obtain matching X Server display fonts (if necessary).

HPD and TFM files

PCL5 uses HPD and TFM files. The HPD files contain a list of fonts available for the printer and each font refers to a TFM file. The HPD file is an ASCII file, which can be edited, but the TFM file is a binary file, which cannot be edited. Even though TFM files are binary and uneditable, you can perform string operations to read some specific keywords from these files. Oracle Reports recognizes the font name that is in the TFM files and not the one specified in the HPD file. The font vendor should provide TFM files and new fonts should be added to the HPD file for your printer when installed.

Tk2Motif.rgb (UNIX only)

This file contains resource settings for all Oracle Motif tools based on Oracle Toolkit. For font specific resource settings, `Tk2Motif*fontMapCs` and `Tk2Motif*fontList` are used.

FontMapsCs governs the base character set of fonts that the application will use, which are on the X-window display.

If `Tk2Motif*fontMapCs: iso8859-2=EE8ISO8859P2`, then the `NLS_LANG` should be `EE8ISO8859P2` and only fonts with encoding as `iso8859-2` will be used for the application. If the system does not find any fonts with the above encoding, it will fail with a `REP-3000` error.

`Tk2Motif*fontList` specifies the default system font that will be used by the application. The following means that the Helvetica font with medium weight and normal width of size 12 will be used:

```
Tk2Motif*fontList: -*-helvetica-medium-r-normal-*-120*
```

The syntax for the above entries can be found in `Tk2Motif.rgb` file as comments.

4.3.1 File Searching

The criteria for searching files is dependent upon the type of file and the various environment variables defined.

Table 4–1   File information

Filename	Type	Description
uiprint.txt	UNKNOWN	Printer configuration file
uifont.ali	FONTALIAS	Font aliasing file
PPD	PPD	PostScript printer definition file
AFM	AFM	Adobe Font metrics file
HPD	HPD	HP glue file
TFM	TFM	HP glue file

Oracle Reports will first look for the variable in TK90_*type*, then in the ORACLE_ *type*, and then in the global directory. For instance, the PPD files are searched for in the directory specified by TK90_PPD, then in ORACLE_PPD, and then in *ORACLE_ HOME*/guicommon9/tk90/admin/PPD.

For example, looking for uiprint.txt, Oracle Reports will first look at the environment variable TK90_UNKNOWN, then look at ORACLE_UNKNOWN, and then in the default directory.

Environment Variables:

REPORTS_NO_DUMMY_PRINTER: If this variable is defined to any value, Oracle Reports will use screen fonts instead of the printer fonts.

4.4 Font Aliasing

Font aliasing is a mechanism in Oracle Reports that allows a font or its associated attributes like style, weight, width, size and character set to be mapped to another desired font or its associated attributes. Its primary use is when applications are ported from one platform to another and the font associated with some or all of the objects in the layout on the source platform do not exist on the target platform. In such cases font aliasing will be helpful as the nonexistent fonts can be mapped to another available one producing the required results. For example, when moving from MS Windows to Motif one would use font aliasing to map the MS Arial to a font available on Motif, such as Helvetica.

4.4.1 Specify Aliasing Information

The file that contains mapping specifications is uifont.ali. To include any new or changed mapping rules, you must edit this file.

The general format is,

```
"Original_font"  = "Font_to_be_aliased"
```

where `Original_font` is the font name or its other attributes that will be mapped to the font name or attributes of `Font_to_be_aliased`.

The fonts along with their attributes can be described as:

```
Face.Size.Style.Weight.Width.CharSet =
Face.Size.Style.Weight.Width.CharSet
```

The *Face* must be the name (string/identifier) of a font face like Courier. The *Style, Weight, Width,* and *CharSet* may either be a numeric value or a predefined identifier/string. For example, both Plain and 0 are valid *Style* values and refer to the same style. The *Size* dimension must be an explicit size, in points.

These attributes take effect for font aliasing, font subsetting, and font embedding.

For example, in the case of font subsetting it is:

```
Font name = Filename.ttf
Face.Size.Style.Weight.Width.CharSet = Filename.ttf.
```

The following is a list of recognized names and their numeric equivalents:

Table 4–2   Style names and their numeric equivalents

Style name	Numeric equivalent
Plain	0
Italic	1
Oblique	2
Underline	4
Outline	8
Shadow	16
Inverted	32
Blink	64

Table 4–3   Weights and their numeric equivalents

Weight name	Numeric equivalent
Ultralight	1
Extralight	2
Light	3
Demilight	4
Medium	5
Demibold	6

Table 4–4   Widths and their numeric equivalents

Width name	Numeric equivalent
Ultradense	1
Extradense	2
Dense	3
Semidense	4
Normal	5
Expand	7
Extraexpand	8
Ultraexpand	9

Styles may be combined; you can use the plus sign (+) to delimit parts of a style. For example:

```
Arial..Italic+Overstrike = Helvetica.12.Italic.Bold
```

This mapping indicates that any Arial that has both Italic and Overstrike styles will be mapped to a 12-point, bold, italic Helvetica font.

For multibyte language support, you must alias a character set with a CID font (CID Fonts) from the Asian font pack from Adobe. For example, in your Japanese report you have aliased a multi-byte `Shift-JIS` characterset be aliased to HeiseiKakuGo-W5-Acro CID font with the following entry: .....JA16SJIS = "HeiseiKakuGo-W5-Acro"

All strings are case-insensitive in mapping. Font faces are likely to be case-sensitive on lookup, depending on the platform and surface. As a result, take care with the names used. For example, if the font name `arial` is used on the left-hand side (the original font), all layout objects with fonts such as `arial` or `Arial` are mapped to the aliased font.

4.4.2 Font Aliasing Mechanism

For font aliasing, Oracle Reports searches for entries under the related section in the alias file that matches the original font attributes given in the report. Refer to Font Alias File Sections for more information about the sections of the font alias file. If an exact match is found, Oracle Reports maps the original font on the left to the target font on the right.

For example:

```
Arial.8.Italic.Medium.Normal.WE8ISO8859P1 = Helvetica.12.Plain.Light.Normal.WE8ISO8859P1
```

If an Arial font with all of the attributes listed on the left is found, it will be mapped to a Helvetica font with all of the attributes listed on the right.

Any field can have a blank entry, which means it will be matched regardless. For instance:

```
Arial..... = Helvetica.12.Plain.Light.Normal.WE8ISO8859P1
```

In this case, all of the Arial fonts, irrespective of size and other attributes, are mapped to Helvetica with size 12, style Plain, weight Light, having Normal width under character set `WE8ISO8859P1`.

Another way to specify an aliasing rule is:

```
Arial = "OCR B"
```

This method will preserve the other attributes of the present font but will change the font name to `OCR B`. You need to be certain in such cases about the availability of mapped fonts with the attributes of other fonts. For example, in this rule the Arial font with style Italic might be mapped to the `OCR B` font with Plain style because the `OCR B` font does not have the Italic style present.

After a mapped font is read from the alias file (`uifont.ali`), Oracle Reports looks for the font following the font lookup procedure, which is described in Font lookup. If the mapped font is found on the system, then Oracle Reports uses this font. Otherwise, it looks for the original font in the system.

Font attributes are searched for with the font face, size, style, weight, and width under the specified character set.

In Oracle Reports, fonts for the Web Source view and PL/SQL editor can be mapped by providing a mapping specification in the [RWBUILDER] section. This feature is mainly intended for supporting Unicode fonts in these editors.

4.4.3 Font Alias File Sections

The uifont.ali file can be found in the following locations for Oracle Reports:

```
ORACLE_HOME\tools\common90 (Windows)
ORACLE_HOME/guicommon9/tk90/admin (UNIX)
```

The alias file consists of various sections which contains font mapping instructions for a particular area. Since Oracle Reports looks in specific sections for specific purposes, it is crucial that you place your mapping entries in the appropriate section for what you are trying to accomplish.

Table 4–5   Font mapping file sections

Section name	Description
Global	Applies everywhere.
Printer	Only applies to printer output.
Printer:PostScript1	Applies to PostScript Level 1 printers.
Printer:PostScript2	Applies to PostScript Level 2 printers.
Printer:PCL5	Applies to PCL 5 printers.
Display	Only applies to the display (the screen).
Display:Motif	Applies only to the Motif display.
Display:CM	Applies only to character-mode display.
PDF	Used for font aliasing (from Oracle Reports 6*i*) and multibyte language support (from Oracle Reports)
PDF:Embed	(Oracle Reports only) Used for Type 1 font embedding.
PDF:Subset	(Oracle Reports only)

Table 4–5   Font mapping file sections

Section name	Description
RWBUILDER	(Oracle Reports only) Fonts for the Web source and PL/SQL editor can be mapped in this section.
printer_name	A section for a specific printer, such as:
	`[Printer:PostScript1:2op813a]`

> **Note:**   In the Web Source view of the Report Editor, the following languages may appear garbled: Japanese, Thai, Arabic, and Hebrew. To work around this issue, you can set the font names for Reports Builder in `uifont.ali` as follows:
>
> `[rwbuilder]`
>
> `.....ja16sjis="MS Gothic"`
>
> `.....ar8mswin1256="Courier New"`

Order of precedence

When aliasing a particular font, only one section is read based upon the context in which the font is used. Hence, if three sections apply, only one is read. For example, suppose you have three sections, `[Printer]`, `[Printer:PostScript]`, and `[Printer:PostScript:2op813a]`. When generating output, if the printer is `2op813a`, only the mapping rules in section `[Printer:PostScript:2op813a]` are read. For printers other than `2op813a`, Oracle Reports would use the `[Printer:PostScript]` section.

The more specific sections of the alias file take precedence over the more general sections. For example, a specific printer section, such as `[Printer:PostScript1:2op813a]` would take precedence over the `[Printer:PostScript1]` section, which would take precedence over the `[Printer]` section, which would take precedence over the `[Global]` section.

The `uifont.ali` file is the configuration file controlling all of the Oracle Reports PDF font enhancements. It can be found in the *ORACLE_HOME*\tools\common90 (Windows) directory and in the *ORACLE_HOME*/guicommon9/tk90/admin (UNIX) directory. The `uifont.ali` file is text readable, i.e., you can edit it with a standard text editor. Exercise caution when editing the file. The `uifont.ali` file

should be saved as a text file with no formatting or special characters that may corrupt the file.

> **See Also:** Chapter 6, "Using PDF in Oracle Reports"
>
> For more information on the various sections in the `uifont.ali` file.

4.4.4 Font Aliasing File Verification

`fontchk90` is used to verify whether the `uifont.ali` file is correct and can be found in the *ORACLE_HOME*/bin directory. It is always advisable to run this utility on the modified `uifont.ali` file to catch any errors:

```
fontchk90 [filename]
```

where *filename* is the name of the `uifont.ali` file. If you don't specify any file name, it will check the default file based on the environment variables.

If the alias file has errors, the utility returns an error message along with the file on which the error was found. For example:

```
Parsing font alias file "/home/oracle/guicommon9/tk90/admin/uifont.ali"
Ms san serif

Error at line 85: Invalid font specification
Parse of font alias file failed
```

The above error indicates that there is a syntax error in `uifont.ali` in the mapping rule for MS San Serif font on line 85.

4.5 Troubleshooting Font Issues

How to check whether the desired font is used in a PostScript file:

PostScript files have a list of fonts, which is created after reading the PPD file. If you examine the PostScript file, you can check the fonts by looking for the following tags:

- `DocumentNeededResource` has the list of fonts referenced in the PPD file.

- `DocumentSuppliedResource` has the list of fonts for which the PostScript driver was able to find the AFM file.

- %%Page paragraph before the field's %IncludeResource:font has the font name which will be used for the field.

For PCL output files, you can check whether a particular font was used or not. Depending on this information the font settings in Oracle Reports or the printer can be modified.

Example:
The test results below are based on a Lexmark Optra printer. The fonts and their numbers as well as the control commands are examples and may vary with other printers.

Font info   The Lexmark has a small menu with the option of printing all available fonts (PCL Emulation Fonts). This includes both resident fonts (defaults) and Flash fonts (installed on the printer separately)

Table 4–6   Sample font information

Font name	Style	Weight	Example output
R0 Courier	0	0	... <ESC>(<symset><ESC>(s0p<pitch>h0s0b4099T...
R39 Courier Bold	0	3	... <ESC>(<symset><ESC>(s0p<pitch>h0s3b4099T...
R40 Courier Italic	1	0	... <ESC>(<symset><ESC>(s0p<pitch>h1s0b4099T...
R55 Century Schoolbook Roman	0	0	... <ESC>(<symset><ESC>(s1p<point>v0s0b24703T ...

Table 4–7   Sample Flash font information

Font name	Symbol set	Style	Weight	Example output
F2 OCR-A	0O	0	0	... <ESC>(0O<ESC>(s0p<pitch>h0s0b4200T ...
F3 OCR-B	1O	0	3	... <ESC>(1O<ESC>(s0p<pitch>h0s0b4206T ...

In these examples, there are many more fonts and each font has its own code. OCRB for example has code 4206. This number is important later on.

Creating output
When having problems getting the correct font, simplify the report and thereby the output. This can be done by creating a straightforward report using select

`sysdate from dual` as the query and limiting the number of fonts. This will avoid long runs and create much smaller output files.

The output file

The resulting PCL-file is a binary file but is reasonably readable in the VI editor. The first small part and the end part is binary, but the middle part is readable and contains data that can be interpreted.

Verifying the output file

The only interesting information is in the readable, middle part of the file. Find the text (this is the text displayed in the reports output) and check out the part preceding the text.

It looks like this:

```
....;SD1,14,2,0,3,10.34,5,0,6,0,7,4099;LB here is your text
```

In the example above, the font is selected with code 4099. For the Lexmark printer, this is selecting Courier.

In one example, the font OCR-B (code 4206) was needed. The font did not come out until that specific code was generated just before the selected text. It looks like this:

```
....;SD1,14,2,0,3,8.57,5,0,6,0,7,4206;LBThis is OCRB font....
```

Problems & answers

If the output file contains the correct code, but the font does not appear on the printer, the printer probably does not have the font available. This will also occur if the code in the output file (deduced from TFM file) is not the same as the one the printer is expecting. On the Lexmark printer, the font was replaced by the default font on the printer.

If the output file does not contain the code for the font, Reports Builder did not generate the code to the output file. Check for the HPD and TFM file.

Environment variables

`DEBUG_SLFIND` can help you ascertain which of these files was used. With reference to the fonts, you can find the list of AFM/TFM files the application looked at after reading the printer definition file and which font files it read after the aliasing. In this manner, you can also determine whether a font is mapped or not. Usually the order of file reading will be as follows.

- First read the printer definition file.

- Read all the associated font files for the font supplied by this printer definition file

- Read in the alias file

- If there is a mapping of file then read in font information files for those fonts and finally again read the AFM file for the fonts that are used in generating the output.

`TK_DEBUG_POSTSCRIPT` will affect PostScript output. It can be set to any combination of these strings:

- Functions list each toolkit function called in comments in the PostScript output.

- Long produces long, slow, intelligible PostScript.

- Memory displays memory usage at the bottom of each page.

Any of the options can appear in the environment variable, abbreviated down to one letter. You can set it to any combination of these, separated by "/". This variable is case insensitive. For example, `Func/L/Mem` would give you all three options.

Note that the output that results from using this variable will not be supported by Oracle for customer use. It exists for diagnostics purposes only.

Repairing fonts not appearing correctly in Web source view

Text in the user interface of Reports Builder, such as the window title, uses fonts taken from the system resource files for the current language. These system resource files are supplied with the Oracle Reports installation. In Oracle Reports, you can map these fonts in the `[RWBUILDER]` section of `uifont.ali`. If found, the mapped font is used instead of the original font; if not, Oracle Reports uses the original font.

> **Note:** The mapped font needs to be a fixed-width font.

In the Web Source view of the Report Editor, the following languages may appear garbled: Arabic, Central European languages, Cyrillic, Greek, Hebrew, Japanese, Thai, and Turkish. To work around this issue, you can set the font names for Reports Builder in `uifont.ali` as follows:

```
[rwbuilder]
.....AR8MSWIN1256="Courier New"
.....CL8MSWIN1251="Courier New"
.....EE8MSWIN1250="Courier New"
```

```
.....EL8MSWIN1253="Courier New"
.....IW8MSWIN1255="Courier New"
.....JA16SJIS="MS Gothic"
.....TH8TISASCII="Andale Duospace WT"
.....TR8MSWIN1254="Courier New"
```

You can download a copy of the Andale Duospace WT (fixed-width) font from Metalink, (`http://metalink.oracle.com`). The ARU number is 2638552.

Limitations

On Windows:

- For Unicode, Oracle Reports relies on the Windows operating system for the font and input method for different languages. To enter and display text in a particular language, you must be running a version of Windows that supports that language. Font support is limited, but not restricted, to the Windows NT operating system font support.

- In JDK, a bug causes the bold Korean font to appear incorrectly. OracleAS Reports Services uses the JRE and therefore all bold Korean strings in graphs within reports show up incorrectly.

- Windows NT 4.0 and later versions, provides True Type Big Fonts. These fonts contain all the characters necessary to display or print multilingual text. If you try to type, display, or print multilingual text and see unexpected characters, you are probably not using a big font. Big fonts provided by Microsoft under NT 4.0 and later, are as follows: Arial, Courier New, Lucida Console, and Lucida Sans Unicode. Arial Unicode MS, which is a true big font, is available from Microsoft site for download.

- Wingdings fonts may not appear when NLS_LANG is UTF8.

  The only Wingdings fonts available when using UTF8 are the characters between ASC 32 and 127. ASC 252 would display a blank because it is not supported by UTF8.

  Any of the following font sets would provide a reasonable work around.

  - Webdings - chr(97)
  - Wingdings2 - chr(80)
  - Wingdings2 - chr(87)

On UNIX:

- AFM support is only for single byte PostScript file generation except for the Japanese encoding. The encoding schemes supported for the AFM files are AdobeStandardEncoding, ExtJIS12-88-CFEncoding, FontSpecific, HRoman, ISOLatinHebrew, JIS12-88-CFEncoding, and JIS12e-88-CFEncoding.

  AFM version that is supported is 2.0

- X11 does not support the underline font attribute. Output to file should work according to steps given below.

- In JDK, a bug causes the bold Korean font to appear incorrectly. OracleAS Reports Services uses the JRE and therefore all bold Korean strings in graphs within reports show up incorrectly.

- PostScript printing will not load the fonts to the printer. So for the desired fonts to appear in the printed output, it is necessary that those fonts should be installed on the printer.

- For PCL output, only TFM font formats are supported.

- The display system on UNIX (for example, X11) is totally independent of any application or printer. There is no direct connection between printing and displaying. There can be a font displayed on your screen that is not printed.

  Display and printer fonts are somewhat similar but have more differences than similarities.

  X fonts (display fonts) are bitmap display glyphs, which are displayed on an X terminal by an X Server.

  Printer fonts are PostScript fonts (mathematical descriptions of fonts, not bitmaps) that are present in a PostScript printer and are generated by a PostScript Interpreter on that printer.

- Font size changes after applying a template.

  Creating a template with font set to Times New Roman size 10 (for all fields) and making the report use this template, makes the Paper Design view of the Report Editor display a different font size.

  The reason for this behavior is that defaulting couldn't fit the layout into the desired area.

  First it reduced the size of text fields and then reduced the size of the fonts. This is much better than wrapping the fields and keeping the template font size.

Also, for templates, the font chosen may be different to that in the template since it matches first on the character set. So if the template font doesn't support the current character set, the font will change to one that does. This is mostly visible if you have an English template, which you use in a Hebrew/Arabic environment.

Common Problems

Problem: Letters are truncated from the right margin on printed label reports.

You have printed a mailing label report on a Windows machine and notice that the last letter, or last few letters, on each line are being truncated. The letters are not missing when you preview the report. You have tried changing the page formatting and font settings, but this has failed to resolve the problem.

Solution: If the report displays correctly using a DESTYPE of Preview, this is not a problem with the printer driver. The problem may be occurring due to the frame properties.

If a frame around the layout objects has a Horizontal Elasticity setting of Fixed and the data exceeds the frame size, it can cause this truncation of data.

Try testing the results after changing Horizontal Elasticity to Expand or Variable.

Problem: When generating to file as HTMLCSS, a column is dropped off in the output.

You are generating a report to an HTMLCSS file format and it appears to be fine in the Paper Design view of the Report Editor. When you click on the newly created file it comes up in your browser, but the last column is missing from the report output.

If you re-run the report again, it still looks fine in the Paper Design view and the column is there as it should be. Clicking on the file again appears to have the column dropped off and missing from the report output. PDF appears fine in Paper Design view and the Adobe Acrobat reader.

Solution:

1. Quit Oracle Reports and any other open applications.

2. Choose **Windows Control Panel > Display > Settings**.

3. Set your fonts to be Small Fonts, click **Apply** button and then click **OK** to reconfigure your Windows font settings.

4. Reboot your computer in order for the new font settings to take effect.

5. You can now go back into **Windows Control Panel > Display > Settings** to verify that you have small fonts as a default for your system.

When you click on the HTMLCSS file your browser shows the report correctly with all of the columns intact.

When viewing HTMLCSS files with your browser, it is recommended to have Small Fonts as the default setting for your Windows system.

If you have Large Fonts as your default, your HTMLCSS file may not display correctly.

Problem: How to choose bitmap fonts sizes of less than 8 point in Reports Builder.

Solution: There are times when a font size of 6 or less is required for reporting purposes. Keeping in mind that font mapping and sizing is actually a product of operating system font files and driver/printer specifications, it is possible to change many fonts to minimal sizes such as 6 or less.

Oracle Reports typically allows fonts to be downsized to a size of 8. This is accomplished by opening a report in Reports Builder, going to the Layout Model view, and selecting the report objects that you wish to change. Once the object is selected, go to the font size list next to the font picker and select your font size.

Typically, your size will be limited to a range from 8 to 72 for True Type fonts, less for other fonts.

You can enter a size smaller or larger than the sizes in the list. To do this, again select the object, place your cursor in the font size field, press Delete to remove the current size number, enter the font size you desire, and then press the TAB key. The change takes effect immediately.

Once again, keep in mind that not all font sizes are possible. Also, some combinations of fonts and attributes are not practical. Simply having the ability to choose a font size does not mean that the font will be legible when printed. Fonts that involve small sizes, combined with bold, italic, or other attributes, may also present legibility problems when printed or displayed due to the limitations of the printer driver, printer, font metrics, language, code sets, NLS_LANG, and, of course, human eyesight.

Problem: The report output font size is different in Windows and UNIX.

A simple report designed on Windows NT uses the Arial and a font size of 8. This report was ported to Sun Solaris and was found to have a different font size in the

output on Solaris. In the UNIX environment, the report is uses the Helvetica font and a font size of 9. The Arial font has been mapped to the equivalent font, Helvetica, on Solaris using `uifont.ali`.

Solution:

1. First look for the font size available for Helvetica on the UNIX system by either using the `xlsfont` command or any other UNIX font utility.

2. You should map variable sized fonts on Windows to variable sized fonts on Solaris. For example, modify the mapping for MS Windows Arial.8 = Helvetica.8 (assuming that size 8 is available for Helvetica on the UNIX system) and make sure `uifont.ali` is in the correct directory (see font mapping).

It's probable that the Helvetica font installed on your machine is bit mapped (rasterized) and so it doesn't automatically scale to any arbitrary size. If so, you need to install a scalable Type 1 font, which should allow you to choose any point size.

There may always be differences between fonts on different systems even if the fonts installed are the same because the font configuration files may be different on these systems.

Problem: When printing, fonts are replaced by non True Type fonts. In the Paper Design view, the fonts are fine.

Solution: Check the printer settings (advanced) and make sure that it doesn't say:

```
True Type Font: Substitute with Device Font
```

UNIX

Problem While running Oracle Reports on X-windows emulators, fonts installed on UNIX do not appear in the font lookup box.

Solution: On X-windows emulators, where the font path is usually a font directory on the local machine, the fonts that were installed on will not be available and only the fonts in the local font directory will be used by the Oracle Reports font lookup box. In such cases, you should start a font server on a remote machine where the fonts were installed and point the font path entry to this font server. For starting the font server and setting the font path entry, consult the system manual and X-windows emulator help.

For finding the font path or font server that is currently being used, use the UNIX command xset -.

4.6 Font Types

This section discusses the fonts and character sets relevant to Oracle Reports:

- Character Sets
- Unicode
- Type1 Fonts
- TrueType Fonts
- Barcode Fonts
- CID Fonts

4.6.1 Character Sets

The character set component of the NLS environment variables specifies the character set in which data is represented in your environment. When data is transferred from a system using one character set to a system using another character set, it is processed and displayed correctly on the second system, even though some characters might be represented by different binary values in the character sets.

If you are designing a multilingual application, or even a single-language application that runs with multiple character sets, you need to determine the character set most widely used at runtime and then generate with the NLS environment variable (NLS_LANG) set to that particular character set.

If you design and generate an application in one character set and run it in another character set, performance can suffer. Furthermore, if the runtime character set does not contain all the characters in the generate character set, then question marks appear in place of the unrecognized characters. Portable Document Format (PDF) supports multibyte character sets. There might be situations where you create an application with a specific font but find that a different font is being used when you run that application. You would most likely encounter this when using an English font (such as MS Sans Serif or Arial) in environments other than Western European. This occurs because Oracle Reports checks to see if the character set associated with the font matches the character set specified by the language environment variable (NLS_LANG). If the two do not match, Oracle Reports automatically substitutes the font with another font whose associated character set matches the character set

specified by the language environment variable. This automatic substitution assures that the data being returned from the database gets displayed correctly in the application. Note: If you enter local characters using an English font, then Windows does an implicit association with another font. There might be cases, however, where you do not want this substitution to take place. You can avoid this substitution by mapping all desired fonts to the `WE8ISO8859P1` character set in the font alias file (`uifont.ali`).

4.6.2 Unicode

Unicode is a global character set that allows multilingual text to be displayed in a single application. This enables multinational corporations to develop a single multilingual application and deploy it worldwide. Oracle Reports provides Unicode support. If you use Unicode, you are able to display multiple languages, both single-byte languages such as those spoken in Western Europe, Eastern Europe, Bidirectional Middle Eastern, and multibyte Asian languages such as Chinese, Japanese, and Korean (CJK) in the same application. Use of a single character set that encompasses all languages eliminates the need to have various character sets for various languages. For example, to display a multibyte language such as Japanese, the `NLS_LANG` environment variable must be set to the following:

```
NLS_LANG=JAPAN_JAPANESE.JA16SJIS
```

To display a single-byte language such as German, `NLS_LANG` must be set to the following:

```
NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1
```

The obvious disadvantage of this scheme is that applications can only display characters from one character set at a time. Mixed character set data is not possible.

With the Unicode character set, you can set the character set portion of `NLS_LANG` to UTF8 instead of a specific language character set. This allows characters from different languages and character sets to be displayed simultaneously. For example, to display Japanese and German together on the screen, the `NLS_LANG` variable must be set to one of the following:

```
NLS_LANG=JAPAN_JAPANESE.UTF8
NLS_LANG=GERMAN_GERMANY.UTF8
```

Oracle Reports relies on the operating system for the font and input method for different languages. To enter and display text in a particular language, you must be running a version of the operating system that supports that language. Font support is limited but not restricted to the operating system font. Windows NT release 4.0

and later versions provide True Type Big Fonts. These fonts contain all the characters necessary to display or print multilingual text. If you try to type, display, or print multilingual text and see unexpected characters, then you are probably not using a Big Font. Big Fonts provided by Microsoft under Windows NT release 4.0 and later, are as follows:

- Arial
- Courier New
- Lucida Console
- Lucida Sans Unicode
- Times New Roman

Third-party Unicode fonts are also available.

To enable Unicode support, set the NLS_LANG environment variable as follows:

```
NLS_LANG=character_set.UTF8
```

4.6.3 Type1 Fonts

PostScript font formats Adobe Type 1 fonts are stored in two common formats: `.pfa` (PostScript Font ASCII) and `.pfb` (PostScript Font Binary). These contain descriptions of the character shapes, with each character being generated by a small program that calls on other small programs to compute common parts of the characters in the font. In both cases, the character descriptions are encrypted. Before such a font can be used, it must be rendered into dots in a bitmap, either by the PostScript interpreter, or by a specialized rendering engine, such as Adobe Type Manager, which is used to generate low-resolution screen fonts on Apple Macintosh and on Microsoft Windows systems.

The Type 1 binary files(.pfa, .pfb) contain character information while the metric (.afm - Adobe Font Metric or .pfm - Printer Font Metric) files contain the metric information to form the character. These metrics files are ASCII files with a well-defined easy-to-parse structure.

4.6.4 TrueType Fonts

The personal computer brought about a need for scalable font technology, thought to be an important part of any future operating system. TrueType is this scalable font technology that enables you to view the same output without the jagged aliasing caused by scaling that is apparent when bitmapped fonts are used.

This technology involves two parts:

- The Rasterizer
- TrueType fonts

The Rasterizer is an application that is included in both Windows and Macintosh operating systems. It acts as an interpreter and translates the font information into a form that the video display can render.

The TrueType fonts themselves contain information that describes the outline of each character in the typeface. Higher quality fonts also contain hinting codes. Hinting is a process that makes a font that has been scaled down to a small size look its best. Instead of simply relying on the vector outline, the hinting codes ensure that the characters line up well with the pixels so that the font looks as smooth and legible as possible.

Adobe wanted both Apple and Microsoft to license its PostScript code, which was capable of handling this role, but both companies were concerned about having a third party control key parts of their operating systems. Apple and Microsoft agreed to a cross-licensing and product development deal, with Microsoft creating a PostScript-style graphics engine and Apple creating a font system. Apple developed what was to become TrueType, which proved superior to other competing technologies on performance and rendering quality. Apple and Microsoft announced their strategic alliance against Adobe, where Apple would do the font system, Microsoft the printing engine. Apple released TrueType in March 1991 and the first TrueType fonts:

- Times Roman
- Helvetica
- Courier

Microsoft introduced TrueType into Windows with version 3.1 in early 1992. They created a core set of fonts:

- Times New Roman
- Arial
- Courier

Both Apple's and Microsoft's TrueType fonts showed that scalable fonts could generate bitmaps virtually as though each size had been designed by hand.

4.6.5 Barcode Fonts

Barcode fonts can be quite confusing. Some industries have chosen a specific barcode type. If this is what you need, then using the appropriate barcode font should work. For example, if you are interested in putting barcode on retail packages or books, the choice of a barcode is simple. Retail packages in North America use the UPC-A bar code. European retail articles use the EAN barcode .

All book ISBN numbers use the Bookland barcode (an EAN 13 bar code with a 5 digit supplement). Fonts are one way to obtain barcode, but not the only way. Oracle Reports offers another solution for producing barcodes using a Java barcode bean. The Java barcode bean is capable of creating barcodes based on the most popular barcode types.

4.6.6 CID Fonts

Character IDentifier (CID) fonts are a format of composite (multibyte) Type1 fonts used to better address the requirements of Far East markets. Adobe developed the CID-keyed font file format to support large character set fonts for use with PostScript. It is the ideal format for Chinese, Japanese, or Korean fonts and can also be used for roman fonts with very large character sets. CID-keyed refers to the character identifier (CID) numbers used to index and access the characters in the font. A CID (character identifier) font consists of a large font file containing all the character outlines and a small CMap file that contains a list of characters, encodings, and character identifiers. The combination of the font file and the CMap file yields a font that is a specific character set and encoding information. Each CID font can support many character set and encoding combinations.

5

Printing on UNIX with Oracle Reports

Oracle Reports provides a rich set of features out of the box for printing on various platforms. Printing on UNIX requires some setup and configuration to create the proper printing environment. This chapter provides information about printing on UNIX with Oracle Reports. In particular, it covers:

- UNIX Printing Overview
- Setting Up a Printer on UNIX
- Configuring the Printing Environment
- Printer-Related Files
- NLS Support
- Debugging Options
- Frequently Asked Questions

5.1 UNIX Printing Overview

This section explains how to print from Oracle Reports on UNIX and highlights the key differences between the UNIX and Windows platforms. It also explains the operating system requirements for any application to print successfully.

- General Printing Mechanism
- Oracle Reports Printing Mechanism on UNIX and Windows
- Printing Support

5.1.1 General Printing Mechanism

In trying to understand how printing works for Oracle Reports on UNIX, it is useful to have the Microsoft Windows printing mechanism as a reference point. Microsoft Windows provides an application level API that supports different types of printers based on the installed printer drivers. Applications can interact with various printer drivers through these standard APIs. For example, to change the paper margin, an application needs to call the appropriate Microsoft Windows API method, which conveys the desired changes to the printer driver. On Microsoft Windows, printer drivers are printer specific, i.e., you need to install a specific printer driver for a printer. These printer drivers know how to communicate to the printer and provide services to applications that need to send output to the printer. Applications can access the printer properties, change their properties, and perform printing through these standard APIs.

Motif and character-based UNIX operating systems do not have their own standard interface to printers as does Microsoft Windows. Individual applications are responsible for sending their output in a streamed file to the printer and adhering to the specifications of the printer. On UNIX platforms, Oracle Reports output must be formatted properly (e.g., PostScript or PCL) before sending it as a stream to the printers. To print on UNIX, Oracle Reports mimics the behavior of the Microsoft Windows printer drivers internally. The next section describes more precisely how this mechanism works on UNIX.

5.1.2 Oracle Reports Printing Mechanism on UNIX and Windows

The diagrams below depicts the differences between Oracle Reports printing on UNIX and on Microsoft Windows.

Figure 5–1   Oracle Reports printing on UNIX

Figure 5–2   Oracle Reports printing on Microsoft Windows
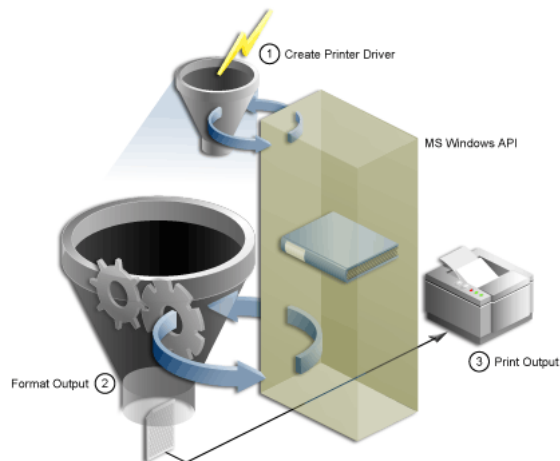
To support printing on UNIX, Oracle Reports internally creates logical printer drivers. A logical printer driver simulates the behavior of Microsoft Windows

printer drivers and provides a printing service interface for Oracle Reports on UNIX. Through the logical printer driver, Oracle Reports can access the printer properties and perform printer-related operations. These logical drivers:

- Support PostScript and PCL printing specifications, which are the most popular printing standards.

- Read the printer description files (e.g., PPD or HPD) to get the printer descriptions.

- Embed the various printer commands in the generated PostScript or PCL output. For example, to change paper margin, the logical printer driver needs to write the corresponding printer commands in the generated output. These commands differ depending on whether you use a PostScript or a PCL printer. When the generated PostScript or PCL file is sent to the printer via the printing executable (e.g., `lpr`), the printer interprets the commands in the file and processes them accordingly.

To function correctly, the logical printer drivers require the following input:

- The printer queue name that is used to spool the print request.

- The printer description file that contains the printer properties.

- The driver type required by the specified printer queue, PostScript or PCL.

You provide this information in a file called `uiprint.txt`. Oracle Reports uses this file to get a list of the printer queue names available for printing. In `uiprint.txt`, you need to specify the printer queue name, the type of driver needed for the queue, the version of the driver, and any special printer description file that the print driver needs for that specific printer (e.g., a PPD file for the PostScript driver). Once this information is available, the internal logical printer drivers are constructed and they use the definition files provided to access the printer properties.

5.1.3 Printing Support

Oracle Reports supports the following printing standards on UNIX:

- PostScript Level 1 and 2

- PCL Level 3

- ASCII (for character mode printing)

The printers you use with Oracle Reports should be compatible with the above versions.

5.2 Setting Up a Printer on UNIX

This section describes:

- Installing a Printer on UNIX

- Verifying the Printer Setup for Oracle Reports

5.2.1 Installing a Printer on UNIX

The installation of a printer queue is slightly different depending upon your flavor of UNIX. Some platforms may have user interface tools to help in the installation. Please refer to your UNIX platform documentation for the steps on adding a printer queue.

The following sample script adds a printer queue on the Solaris 2.6 platform. The domain information `expldomain` and printer names `printer1` and `printer2` are hard coded in this example. The printer is a Xerox DCS model.

```sh
#!/bin/sh
echo "Please enter the Printer Name Either printer1 or printer2\n"
read PRINTER
LOGFILE=/var/adm/config.log
PATH=/usr/bin:/usr/sbin:$PATH
export PRINTER LOGFILE PATH
lpsystem -t bsd expldomain >$LOGFILE 2>&1
lpadmin -p "$PRINTER" -s expldomain!"$PRINTER"  -I any >$LOGFILE 2>&1
mkdir -p /usr/Xerox_DCS /usr/Xerox_DCS/original
chown -R 755 /usr/Xerox_DCS /usr/Xerox_DCS/original
cp /usr/bin/lp /usr/Xerox_DCS/original
mv /usr/bin/lp /usr/bin/lp.Xerox
ln -s /tmp /usr/Xerox_DCS/tmp
echo "$PRINTER" > /usr/Xerox_DCS/printer.db
cp  /usr/local/packages/dc99cc23.txt  /usr/Xerox_DCS
ln -s /usr/Xerox_DCS/dc99cc23.txt /usr/bin/lp
lpadmin -d "$PRINTER"
```

5.2.2 Verifying the Printer Setup for Oracle Reports

To verify that your printer queue installed correctly:

1. Make sure that the PPD or HPD file used with the installed printer queue is available in the following location:

   ```
   ORACLE_HOME/guicommon9/tk90/admin/PPD
   ORACLE_HOME/guicommon9/tk90/admin/HPD
   ```

2. Make sure that the font metrics, AFM or TFM files, installed on the printer are available in the following location:

```
ORACLE_HOME/guicommon9/tk90/admin/AFM
ORACLE_HOME/guicommon9/tk90/admin/TFM
```

5.3 Configuring the Printing Environment

This section explains the various configuration steps to be performed on UNIX after printer installation.

- Editing uiprint.txt File
- Environment Variables
- Print Property Dialog Boxes

5.3.1 Editing uiprint.txt File

As discussed in UNIX Printing Overview, Oracle Reports creates logical printer drivers. To create these internal printer drivers, it needs information from you like the available printer queue, the type of driver to be used with the queue, the version of the driver, and the printer description file. uiprint.txt is the main file for providing this information. It is located in:

```
ORACLE_HOME/guicommon9/tk90/admin
```

uiprint.txt is the printer configuration file and Oracle Reports reads it when it creates the internal printer drivers. You should modify this file for each instance of Oracle Reports.

The format of entries in uiprint.txt is:

```
Printer:DriverType:DriverVersion:PrinterDescription:PrinterDescriptionFile:
```

This one line entry, in prescribed format, in uiprint.txt defines a printer to be used by Oracle Reports. Each line contains five fields separated by colons. Table 5–1 describes each element of the uiprint.txt entry.

Table 5–1   uiprint.txt entry elements

Element	Description
Printer	Specifies the name of the printer (or printer queue), as used with the `lpr` or `lp` command.
	To get a list of all available printers, use the following command:
	lpstat -a
	To check the status of the printer, use the `lpstat` command:
	Solaris
	lpstat -p *printername*
	Linux
	lpstat -p *printername*
	HP-UX
	lpstat -d *printername*
	HP Tru64
	lpstat -p *printername*
	IBM AIX
	lpstat -p*printername*
	No space is allowed after `-p` on IBM AIX.
DriverType	Specifies the type of printer driver used for the printer. The driver can be PostScript, PCL, or ASCII.
DriverVersion	Specifies the version of the driver type that should be used. This can be 1 or 2 for PostScript printers, and PCL Version 5 for PCL.
PrinterDescription	Specifies the description of the printer, for example, the speed and the location of the printer. This information is used for display in the printer-related dialog box.

Table 5–1   uiprint.txt entry elements

Element	Description
PrinterDescriptionFile	Specifies the printer description file to be used with the printer. It can be one of the following types:

PrinterDescriptionFile entries:

- When using a PostScript printer, this entry contains the name of a PPD file. PPD stands for PostScript Printer Description. If Oracle Reports cannot find the specified PPD file, it uses `default.ppd`. Oracle Reports searches for PPD files in:

  ORACLE_HOME/guicommon9/tk90/admin/PPD

- When using a PCL printer, this entry contains the name of an HPD file. If Oracle Reports cannot find the specified HPD file, it uses `ui4.hpd`. Oracle Reports searches for HPD files in:

  ORACLE_HOME/guicommon9/tk90/admin/HPD

- When using an ASCII printer, this entry would be set to none. This field is ignored for all ASCII printers.

Usage Note:

- All the fields in the `uiprint.txt` entry must be filled and every line must end with a colon.

- At least one entry must be defined in `uiprint.txt`. Alternatively, you can set the related printer variables (`TK_PRINTER` and `PRINTER`). Without these, Oracle Reports is unable to perform any printer-related task.

  See Also: Environment Variables

  For more information on printer-related environment variables.

The internal printer drivers provide a drawing surface for Oracle Reports. In addition to using this surface for printing, Oracle Reports uses it internally whenever output is generated to a file. Hence, you need to have a valid entry in `uiprint.txt` or to set one of the printer-related environment variables. To simplify the selection of printers for your users, we recommended that you list all printers accessible to users in `uiprint.txt`.

Example:

Following are two example entries for `uiprint.txt`:

```
colprt14:PostScript:2:RMSC Atrium HPLaserJet5:default.ppd:
```

```
colprt15PCL:5:RMSC 1st Floor HPLaser4:ui4.hpd:
```

5.3.2 Environment Variables

This section lists the environment variables related to printing:

> **See Also:**   Reports-Related Environment Variables
>
> For more information on the various environment variables that can be set in Oracle Reports.

- TK_PRINTER / PRINTER
- TK_PRINTER
- TK_PRINT_STATUS
- REPORTS_NO_DUMMY_PRINTER
- TK90_HPD and ORACLE_HPD
- TK90_PPD and ORACLE_PPD
- TK90_TFM and ORACLE_TFM
- TK90_AFM and ORACLE_AFM

5.3.3 Print Property Dialog Boxes

On UNIX, Reports Builder provides several dialog boxes for printer-related operations.

5.3.3.1 Page Setup dialog box

The Page Setup dialog box enables you to specify how the printed page appears. The available options depend on the type of printer driver being used. The internal printer drivers use this dialog box to get all the information necessary, (e.g., scale, rotation, width, and height) for formatting a page on a printer.

5.3.3.2 Print Job dialog box

Each print job has unique characteristics depending on the printer driver being used. The Print Job dialog box displays just prior to print job execution and prompts you for the print job information required to send the job to the printer.

5.4 Printer-Related Files

This section explains the different printing related files. It gives an overview of these files and also provides information for editing these files for common printing needs.

- Overview of Files
- PPD Files
- HPD Files
- Font Metrics Files
- uifont.ali
- uiprint.txt
- Editing the Printer-Related Files

5.4.1 Overview of Files

Table 5–2 lists files used by Oracle Reports for printing on UNIX.

Table 5–2   Printer-related files overview

File name/extension	Description
.ppd	PostScript Printer Definition file
.hpd	HP glue file
.afm	Adobe font metrics file
.tfm	PCL font metrics file
uifont.ali	font aliasing file
uiprint.txt	printer configuration file

5.4.2 PPD Files

PostScript is Adobe's page description programming language. PPD files define what capabilities a printer has for applications like Oracle Reports. For example, a PPD file might define which paper tray to use, what paper sizes are available, what is the physical dimension of the paper, and what font is available. Currently, Oracle Reports reads the paper sizes and fonts available on the printer as well as its default resolution from this file. In the future, more information may be used, such as memory for proper image partitioning.

The only reason to modify the PPD file is to allow Oracle Reports to recognize newly added fonts or memory. You can also change the `DefaultPageSize` to your preferred page size.

> **Note:** Page sizes, like all PPD entries, are case sensitive. Other entries in the PPD file should generally be left undisturbed.

When you select a printer that is not listed in `uiprint.txt` or change the type of printer to a PostScript type in the Choose Printer dialog box, you are prompted for the PPD file for the printer. You must choose the PPD file for a printer that most closely resembles the printer being used. PPD file names typically bear some resemblance to the printer model name.

In `uiprint.txt`, a PPD file must be specified for each printer. If an invalid PPD file is specified for the current printer (e.g., no PPD file is found or the PPD file format is wrong), Oracle Reports will use `default.ppd` for that printer. You should make `default.ppd` a copy of another PPD file that better reflects the most likely default, local printer.

Oracle Reports includes a common set of PPD files, but sometimes you may need to get specific PPD files for your printers from the vendor. Table 5–3 shows some examples of PPD files that are shipped with Oracle Reports:

Table 5–3   Common PPD files shipped with Oracle Reports

PPD file name	Corresponding Printer
`appl230.ppd`	Apple LaserWriter v23.0
`datap462.ppd`	Dataproducts LZR-2665
`declps32.ppd`	Digital PrintServer 40
`default.ppd`	Default Level 1 PostScript Printer
`hpljet41.ppd`	HP LaserJet 4/4M PostScript 600DPI
`lwntx470.ppd`	Apple LaserWriter II NTX
`nccps801.ppd`	NEC Colormate PS/80
`tkphzr33.ppd`	Tektronix Phaser III PXi v2011.108
`l530_523.ppd`	Linotronic 530
`screenprinter.ppd`	Default PPD file to be used when a printer is not available on UNIX.

If you need a PPD file that is not among those shipped with Oracle Reports, you must do one of the following (in order of preference):

- Ask the printer vendor for the PPD file.

- Download the PPD file from Adobe's Web site

- Copy an existing PPD file and edit it

- Ask Adobe for the PPD specs and write the PPD file.

The PostScript file only has the font information not the font metrics. Oracle Reports refers to the AFM file installed for the font metrics information. The font vendors provide these AFM files. Oracle Reports ships AFM files for some of the most commonly used fonts. The printer must have the required font installed in order to correctly print the PostScript file generated by Oracle Reports.

5.4.2.1 Local customization of PPD files

A PPD file is a static representation of the features of a printer. It contains default factory settings. Once a printer is installed, features such as additional memory, paper trays, and fonts may be added to the device. The task of managing a device is a dynamic issue that requires keeping track of fonts downloaded to disk, error handlers, RAM-based fonts and procedure sets, default device setup, and so forth. This kind of device management is beyond the scope of PPD files. However, there are some provisions for customizing the information contained in PPD files to adapt them to local instances of printers or to specific applications when necessary.

Instead of modifying the original PPD file, another approach would be having a new file having the local customization of certain parameters and refer to the primary file for the remaining information. The local customization file must contain a reference to the primary PPD file in this format:

```
*Include: "filename"
```

where `filename` is the name of the primary PPD file. This referencing allows a system administrator to later replace the primary PPD file without forcing users to edit their local customization files. A file referenced by the `*Include` keyword is treated as though it were in the including (local customization) file.

For example, suppose that the `default.ppd` file is defined as:

```
*PPD-Adobe: "4.0"
*Include: "datap462.ppd"

*% Page definitions
*DefaultPageSize: Letter
```

```
............................
*DefaultPageRegion: Letter
```

The primary PPD file is `datap462.ppd`.

Administrators should change the name of the included file to conform to their site's default printer type.

When a local customization file includes a primary PPD file, there might be several instances of the same keyword in the composite file. Hence, the location of the primary file in the customization file (beginning or end) is important and effects the changes made by the customization file.

5.4.3 HPD Files

HPD files provide functionality for PCL printers that is similar to what PPD files provide for PostScript printers. HPD or HP glue files provide information on what fonts are available for a PCL printer. The HPD file format can be found in the HP PCL5 Developer's Guide.

Just as PostScript has AFM files, every HP font must have an associated TFM file. The font vendor should provide TFM files and new fonts should be added to the HPD file for your printer when installed. For a new font, you should specify the following fields in the HPD file:

```
FONT={fontname}
/tfm={tfm-filename}
```

where

`fontname` is a descriptive name for the font.

`tfm-filename` is the base file name for the TFM file.

If the TFM file isn't specific enough, you can also specify the following after the FONT field:

```
/ptsize={size {size ...}}
```

If the specified font is a bit mapped font but is listed in the TFM file as a scalable font, you can limit the point sizes used by listing the acceptable sizes as follows:

```
/symset={symset {symset ...}}
```

This field limits the supported symbol sets to those listed. See the HP PCL documentation for a list of recognized symbol sets.

Oracle Reports also supports the defaultpaper field for printing to PCL format. This field can be used to set the defaultpaper to be used by the Toolkit. The format of this field is:

```
<defaultpaper={papername}>
```

For example, the following sets the paper name to A4:

```
<defaultpaper=A4>
```

The paper name is case insensitive. If you specify defaultpaper in more than one place, then the last instance of defaultpaper is used. If you specify a paper name that is not supported by the printer, defaultpaper is ignored and LETTER is used as the paper name instead. Similarly, if the paper name is incorrect, then LETTER is used.

5.4.4 Font Metrics Files

Oracle Reports supports two kinds of font metrics files:

- AFM files
- TFM files

5.4.4.1 AFM files

Each AFM files contains the font-related metrics for a single font. The metrics include various font attributes such as style, weight, width, and character set. AFM files and a description of the AFM file format are typically available from the font or printer vendors.

To install the AFM file, just copy it to the AFM file location, which is listed in Verifying the Printer Setup for Oracle Reports. The name of the file must match name of the font without the .afm extension. For example, if the font name is CodedreineunBold, the file name must be CodedreineunBold.

To verify the font name, you can look for the fontname string in the AFM file. Please note that the AFM files are not font files, they are metrics files, which give information on how to properly format the characters for the printer. If you have an AFM file for a font, but the font is not present on the printer, Oracle Reports cannot generate the correct output on the printer because of the font metrics mismatch. You must ensure that the font used to design the report is also available on the printer.

5.4.4.2 TFM files

PCL5 uses HPD and TFM files. The HPD file contains the list of available fonts for the printer and each font refers to a TFM file. TFM files serve the same purpose as Adobe's AFM files, with each file listing information about a single font. The HPD file is an ASCII file, which can be edited, but the TFM file is a binary file, which cannot be edited.

To use a new font in Oracle Reports and have it appear correctly in PCL output, you need the HPD and TFM files for the printer. You can copy an HPD file from an existing one, provided that you ensure it is suitable for your printer. The fonts specified in the HPD file must be available on the printer.

Oracle Reports includes a common set of TFM files. If you need other font metrics files for your printer, you should obtain them from your font or printer vendor. To install the TFM file, just copy it to the TFM file location, which is listed in Verifying the Printer Setup for Oracle Reports.

5.4.5 uifont.ali

The uifont.ali file defines the font aliases used by Oracle Reports. It is an extremely useful tool for cross-platform development because it enables you to define which fonts to substitute when a particular font is unavailable. uifont.ali is located in:

ORACLE_HOME/guicommon9/tk90/admin

To alias a font, use the following syntax:

source_font = destination_font

For each font, you may also specify the following attributes:

face.size.style.weight.width.character_set

Styles may also be combined using a plus sign + to delimit the styles. For example:

```
Arial.Italic+Overstrike = Helvetica.12.Italic.Bold
```

This entry maps any Arial font that has both Italic and Overstrike styles to a 12-point, bold, and italic Helvetica font. Font faces can be case sensitive depending on the platform and the surface, i.e., printer or system.

> **See Also:** Chapter 4, "Managing Fonts in Oracle Reports"

For more font-related information.

5.4.6 uiprint.txt

uiprint.txt provides a convenient way for you to provide details about the printer queue, such as the type of printer driver and the printer description. You should edit uiprint.txt for each instance of Oracle Reports.

> **See Also:**   Editing uiprint.txt File
>
> For more information about uiprint.txt

5.4.7 Editing the Printer-Related Files

The sections that follow describe how to edit the various print-related files.

5.4.7.1 Editing PPD files

In some cases, you may need to change certain attributes in your PPD file. The sections that follow describe some of the attributes that you would commonly need to change.

5.4.7.1.1 Changing the default paper size Suppose that you need the page size to be A4 for some of your reports. On Motif platforms, the printer driver is specified in uiprint.txt and the default page size is not necessarily set to A4. For example, hpljet41.ppd has LETTER as the default page size. Note that the default page size setting for each printer queue is taken from the corresponding PPD file.

To set A4 as the default page size, you would do the following:

1. Edit uiprint.txt to include a PostScript Printer Description file (extension is .ppd) that supports the A4 page size. For example, you might include hpljet41.ppd.

2. As a backup, make a copy of hpljet41.ppd.

3. Add an entry to uiprint.txt:

   ```
   Printer_name: PostScript:1: the printer on floor1:hpljet41.ppd
   ```

4. Edit hpljet41.ppd and change these settings as follows:

   ```
   DefaultPageSize: A4
   DefaultPageRegion: A4
   DefaultImageableArea: A4
   DefaultPaperDimension: A4
   ```

5.4.7.1.2 Changing the printer margin settings To change the margins, you need to modify the ImageableArea section in the PPD file. ImageableArea provides the bounding box of the area in which the printer may print for the page size named mediaOption. There will be one statement for each named page size supported by the device. *DefaultImageableArea provides the mediaOption name of the default imageable area. Since there can be only one default page size, this value should be the same as the value of *DefaultPageSize, *DefaultPageRegion, and *DefaultPaperDimension.

The syntax for defining imageable area is as follows:

```
*ImageableArea mediaOption: "llx lly urx ury "
*DefaultImageableArea: mediaOption | Unknown
```

`ll` stands for lower left corner and `ur` for upper right corner. The bounding box value of *ImageableArea is given as four real numbers, representing the x and y coordinates of the lower left and upper right corners of the region, respectively, in the PostScript language default user space coordinate system. The x and y axes of a given page size correspond to the x and y axes of that page size in the *PaperDimension entry.

The imageable area is defined as the part of the page where the printer may actually make marks. On some printers, the imageable area of a given page size varies as a result of the current resolution, amount of memory, the direction of paper feed, and other factors. In PPD files where the imageable area of a given page size can vary, the imageable area recorded for that page size will be the intersection of all possible imageable areas for that page size. This formula ensures that the available imageable area is never smaller than that shown in the PPD file and all marks made within the imageable area will be visible. It does, however, also mean that the imageable area in the current configuration might actually be larger than the imageable area shown in the PPD file.

The following table contains the option keywords currently registered for `mediaOption`, which designates a given page size on a device:

Table 5–4   mediaOption keywords

mediaOption (paper size)	size (pts)	size (mm)	size (inches)
Letter	612 * 792	215.9 * 279.4	8.5 * 11
Legal	612 * 1008	215.9 * 355.6	8.5 * 14
Ledger	1224 * 792	431.8 * 279.4	17 * 11

Table 5–4   mediaOption keywords

mediaOption (paper size)	size (pts)	size (mm)	size (inches)
Tabloid	792 * 1224	279.4 * 431.8	11 * 17
A3	842 * 1191	297 * 420	11.69 * 16.54
A4	595 * 842	210 * 297	8.27 * 11.69
A5	420 * 595	148 * 210	5.83 * 8.27
B4	729 * 1032	257 * 364	10.12 * 14.33
B5	516 * 729	182 * 257	7.17 * 10.12

Example

To change the margins of an A4 page in the `default.ppd`, you would perform the following steps:

1. Modify the default page from `Letter` to `A4` in the following sections:

```
*% Page definitions
*DefaultPageSize: A4
*PageSize A4: " "

*% These entries set up the frame buffer. Usually used with manual feed.
*DefaultPageRegion: A4
*PageRegion A4: "A4"

*% These provide the physical dimensions of the paper (by keyword)
*DefaultPaperDimension: A4
*PaperDimension A4: "595 842"
```

2. Add the margin definition in the following sections:

```
*% Imageable (writable) areas for each page size, in pixels
*DefaultImageableArea: A4
*ImageableArea A4: "2 2 593 840 "
```

> **Note:**   All PPD entries are case sensitive.

5.4.7.1.3   Adding a new font entry to PPD files   On PostScript printers, Oracle Reports only allows you to use fonts known to be available on the printer. Since printers are

rarely available for personal requests on multi-process operating systems, Oracle Reports gets a complete list of fonts from the PPD file.

When a new font is installed on the printer, a corresponding font entry needs to be added to the printer's PPD file. The format for a font entry is:

```
*Font {fontname}: {encoding} "({version})" {charset}
```

where

`{fontname}` is the Adobe font face name as specified in PostScript.

`{encoding}` is the PostScript encoding name.

`{version}` is the FontInfo version number.

`{charset}` is the Adobe character set.

The `encoding` value has slightly different meanings depending on the font type. If the encoding cannot be determined, the value of encoding may be set to unknown. Fonts are usually re-encoded by applications to provide other encodings; the `charset` value for each font indicates which encodings are possible for that font. For more information, please refer to the PPD specification from Adobe.

When new fonts are added to the printer, the matching AFM files must also be added to the font metrics directory. Oracle Reports requires the AFM files to get the actual font attributes and properly place text on the printed page.

Example

Suppose you add a new font, CodedreineunBold, and need to edit the PPD file to include the new font.

1. In the PPD file, search for:

   ```
   *% Font Information
   ```

2. For the new font, append the following at the end of the paragraph:

   ```
   *Font CodedreineunBold: Standard '(00.1001)" Standard ROM
   ```

5.4.7.1.4 Overriding the printer tray setting The PostScript output generated by Oracle Reports has the tray information embedded into it. The PPD file defines the default tray to be used and is followed by the definitions of valid trays for the printer. To print to a different tray, the `DefaultInputSlot` entry in the PPD file must be updated.

In the PPD file, you should find a section that lists the default tray and the valid input slots. The section typically starts with a line like this one:

```
*OpenUI *InputSlot:    <PickOne>
```

The default tray entry looks like the following:

```
*DefaultInputSlot: Lower
```

The defined slots typically follow the default entry and look like the following:

```
*InputSlot Upper/Multipurpose Tray: "
...
*InputSlot Lower/Paper Cassette: "
```

The section ends with a line like the following:

```
*CloseUI: *InputSlot
```

You can set `DefaultInputSlot` to be any of the values in the list of defined slots.

5.4.7.2 Editing HPD files for PCL printing

In some cases, you may need to change certain attributes in you HPD file. The sections that follow describe some of the attributes that you would commonly need to change.

5.4.7.2.1 Changing the paper size For example, to change the papersize to A4, add the following to the HPD file used:

```
 <defaultpaper=A4>
```

5.4.7.2.2 Adding a new font entry As with PostScript's AFM files, every HP font must have a TFM file in order for Oracle Reports to use it. The font vendor should provide TFM files. You should add new fonts to the HPD file when you install them.

You must specify the following settings in the HPD file for any new font:

```
FONT={fontname}         # {fontname} is a descriptive name for the font
/tfm={tfm-filename}     # {tfm-filename} is the base filename for TFM file
```

> **Note:** The font name entries in HPD files must be unique.

5.5 NLS Support

This section explains multibyte character set printing support from Oracle Reports. It also explains the new PASTA component, which is supported only for Oracle Reports when installed and used in conjunction with Oracle Applications.

- Multibyte Character Set Printing
- Overview of IX and PASTA

5.5.1 Multibyte Character Set Printing

Oracle Reports does not currently support Unicode character sets in PostScript output. As an alternative, you can use Oracle Reports PDF output, which supports multibyte character sets, and print it.

Oracle Reports supports a set of encoding schemes for the AFM files for the multibyte character sets.

> **See Also:** Chapter 4, "Managing Fonts in Oracle Reports"

For more font-related information.

The fonts must be installed on the printer that prints the PostScript report output.

Example

Suppose you build a report and its generated PostScript output contains a Chinese character set. First, you need AFM and PPD files that adhere to the encoding scheme for multibyte character sets. The destination printer must also have the required Chinese fonts installed because the PostScript file generated by Oracle Reports on UNIX does not have fonts embedded in it. The PostScript file contains only the font name and the font metrics taken from the AFM files. If you try to send the report to a printer that does not have the Chinese fonts installed, it will not print the Chinese characters properly.

5.5.2 Overview of IX and PASTA

When installed and used with Oracle Applications, Oracle Reports has an additional utility for font embedding in PostScript output.

For character mode reports, the utility is called PASTA. For bit-mapped reports, a new component called IX is shipped with Oracle Reports. IX allows you to embed the fonts in the PostScript output thereby allowing you to print even if the font is

not installed on the printer. Both PASTA and IX are supported only for Oracle Reports used with Oracle Applications.

When used for character mode reports, PASTA takes tagged character mode output (generated via an appropriate `prt` file) and generates a PostScript rendition of it. IX enables Oracle Reports to print PostScript bit-mapped reports for all character sets, including UTF8, on a PostScript printer. With this functionality, PostScript printing in Unicode as well as all native languages on Solaris is supported. The IX library is turned off by default with the Oracle Reports patch.

Please refer to your *Oracle Applications System Administrator's Guide* for the setup and usage information for IX and PASTA with Oracle Reports. If you are a member of MetaLink (`http://metalink.oracle.com`), you can also get this information from MetaLink notes 189708.1 and 159225.

If you have problems with PASTA, you can use the following technique to isolate the problem:

1. Unset the PASTA environment variable.

2. Try to perform the steps that caused the problem again.

3. If the problem reproduces without the environment variable set, then it should be treated as a normal Oracle Reports printing problem and the diagnostic steps provided in this document should be applied.

   If the problem reproduces only with the PASTA environment variable set, then follow the diagnostic process given in the Oracle Applications documentation.

5.6 Debugging Options

This section explains the different environment variables and techniques available in Oracle Reports for the debugging of UNIX printing problems.

- DEBUG_SLFIND

- TK_DEBUG_POSTSCRIPT

5.6.1 DEBUG_SLFIND

If this environment variable is set, the file-finding routine lists what was searched for and where Oracle Reports searched for it. This information is a tremendous help if your current configuration does not work. You can send the output to a file, `stdout` (for standard output), or to `stderr` (for output to standard error). If you

try to send the output to a file and it cannot be written to, Oracle Reports uses `stderr` instead.

We recommend sending the output to a file because it is faster and the output can be quite large. Sample output from DEBUG_SLFIND is shown below. Notice how the debug information generated helps you identify the various setup issues, such as which PPD and AFM files are being referred to and their location.

You can see all of the following in this output:

- The various environment variables, such as TK90_PPD and TK90_AFM, and their values.

- The resource files, such as the PPD and AFM, and their locations, which helps you to determine if any are the missing.

- The default location of various resource files under *ORACLE_HOME*.

```
slsfindfile(): checking environment variable TK90_PPD(8).
slsfindfile(): environment variable not set
slsfindfile(): checking environment variable ORACLE_PPD(10).
slsfindfile(): environment variable not set
slfpath():     looking up path
               /s/rw/994/solohome/guicommon9/tk90/admin/PPD/
slfexist():    testing /s/rw/994/solohome/guicommon9/tk90/admin/PPD
slfexist():    testing /s/rw/994/solohome/guicommon9/tk90/admin/PPD/default.ppd
slsfindfile():returned
/s/rw/994/solohome/guicommon9/tk90/admin/PPD/default.ppd
slfindfile(): type = 39 (AFM)slfindfile():  name = Courier-Bold
slsfindfile(): checking environment variable TK90_AFM(8).
slsfindfile(): environment variable not set
slsfindfile(): checking environment variable ORACLE_AFM(10).
slsfindfile(): checking ORACLE_HOME environment variable.
slsfindfile(): environment variable set to /s/rw/994/solohome (len=18)
slfpath():     looking up path/s/rw/994/solohome/guicommon9/tk90/admin/AFM/
slfexist():    testing /s/rw/994/solohome/guicommon9/tk90/admin/AFM
slfexist(): testing /s/rw/994/solohome/guicommon9/tk90/admin/AFM/Courier-Bold
slsfindfile():returned /s/rw/994/solohome/guicommon9/tk90/admin/AFM/Courier-Bold
slfindfile():  name = uiprint.txt
slsfindfile(): checking ORACLE_HOME environment variable.
slfpath():     looking up path/s/rw/994/solohome/guicommon9/tk90/admin/
slfexist():    testing /s/rw/994/solohome/guicommon9/tk90/admin
slfexist():    testing /s/rw/994/solohome/guicommon9/tk90/admin/uiprint.txt
slsfindfile(): returned /s/rw/994/solohome/guicommon9/tk90/admin/uiprint.txt
```

5.6.2 TK_DEBUG_POSTSCRIPT

This variable effects the PostScript output generated by Oracle Reports. Table 5–5 shows the settings for this variable.

Table 5–5   Settings for TK_DEBUG_POSTSCRIPT

Setting	Description
Functions (Func)	Function lists each toolkit function called in comments in the PostScript output.
Long (L)	Long produces more intelligible PostScript output but runs much more slowly than normal PostScript generation.
Memory (Mem)	Memory displays memory usage at the bottom of each page.

Any of the options can appear in the environment variable, abbreviated down to one letter. You can set it to any combination of these, separated by "/". This variable is case insensitive. For example, `Func/L/Mem` would give you all three options.

> **Note:**   The PostScript output from this variable is for your own debugging purposes. You do not need to provide this output to Oracle Support for investigation.

5.7 Frequently Asked Questions

This section addresses some commonly encountered problems with UNIX printing.

- Common Printing Error Messages
- PCL Printing Issues
- PostScript Printing Issues
- Font-Related Printing Issues
- Printed Output Issues

5.7.1 Common Printing Error Messages

REP-00177 - Error while running in remote server

REP-1800 - Formatter error

REP-3300 - Fatal error in 'component name

UI-9 - This function call is out of context.

REP-3002: Internal error initializing printer information

Cause:
These errors generally indicate a printer configuration issue.

Solution:
Check the printer queues that have been defined at the operating system level in your setup. You can use:

- `lpc status`
- `lpstat -a`

If a valid printer queue is installed, check for the following:

- `uiprint.txt` must have a valid entry for the printer.
- Oracle Reports must be able to open and read the `uiprint.txt` file:

  The person running the report must have operating system level read permissions on `uiprint.txt`. Oracle Reports must be able to open the `uiprint.txt`. UNIX operating systems do have an open file limit. If you are over that limit, Oracle Reports might not be able to open `uiprint.txt`.

- The printer description files specified in `uiprint.txt` must exist in your installation in:

  ORACLE_HOME/guicommon9/tk90/admin

- The printer specified in `uiprint.txt` must be enabled at the operating system level. A quick test is to try printing any file from the command line using `lp` or `lpr`. If you can print using one of these commands and get the output on the printer, then the printer is enabled.

■ The printer queue and `uiprint.txt` entry syntax must be valid.

  If the printer validation fails, refer to the environment variables `TK_PRINT_STATUS` and `REPORTS_NO_DUMMY_PRINTER` in Environment Variables.

REP-00826 - Invalid printer driver xxx specified by parameter desformat.

REP-00177 - Error while running in remote server (When run through CGI)

Cause:

An invalid value was specified for `DESFORMAT` for the specified report execution mode.

Solution:

The `DESFORMAT` parameter specifies which output format is needed. Valid formats are:

■ For bit-mapped reports, any of the output formats supported by Oracle Reports (PostScript, PCL, PDF, HTML, XML, HTMLCSS) is valid for `DESFORMAT`. You should not give the PRT file names here. While running to a file, the `DESFORMAT` parameter needs to be set to a valid printer queue. Oracle Reports uses the printer definition file associated with the printer to format the output.

■ For character mode reports, `DESFORMAT` sets up the output for ASCII printers and passes escape characters. For running character mode reports, make sure you change the `MODE` parameter to Character and use any valid .PRT file.

Table 5–6 maps the command line options (`DESTYPE`, `DESNAME`, and `DESFORMAT`) to the printer by what you are trying to achieve.

Table 5–6   DESTYPE, DESNAME, and DESFORMAT settings by case

Case	DESTYPE	DESNAME	DESFORMAT
Generating to a file	File	*file_name*.ps	printer name
Printing	Printer	printer name	
Distribute=Yes			printer name
Mode=Character			*file_name*.prt

REP-01800 - Formatter error.

REP-00177 - Error while running in remote server

(When run through CGI)

Cause:

The error indicates that a printer configuration issue has occurred on a UNIX server. Even if there is not a physical printer available on the system, you have to set it up as if there was one.

Solution:

1. Verify that there is a valid entry in uiprint.txt.

2. If you have multiple printer queue entries in uiprint.txt and you need to set the default printer, verify that the environment variable is set to a printer that is listed in uiprint.txt. If the related environment variable is not set, then the first entry in uiprint.txt is used. For more information about printer related environment variables, refer to Environment Variables.

If there is no printer available for your system, refer to Configuring the Printing Environment for alternatives.

Error while printing to a printer with spaces in its name

Cause

If you are on Solaris 2.8 and have printers that have spaces in the names, you may encounter a bug that causes an error resulting from the lpr/lp command including quotes around the printer name.

Solution

To resolve this issue, you must do either one of the following:

- Remove the spaces in the printer's name.

- Install the Solaris 2.8 patch from Sun Microsystems that fixes the lpr/lp command so that quotes can be used in printers names.

  and

- Modify the section of rwlpr.sh that provides the workaround for including quotes, in order to make accessible any printer that has a space in the name. The rwlpr.sh file is located in the *ORACLE_HOME*/bin directory.

Specifically, make the following changes:

```
#either LPR or LP Command was found
if [ -x $PRNCMDPATH ]
 then
   if [ `basename $PRNCMDPATH` = "lpr" ]
   then
    #if [ `/usr/bin/uname -r` = "5.8" ]
    #then
      #$PRNCMDPATH `echo $@ | tr -d "\""`
    #else
      $PRNCMDPATH "$@"
    #fi
  else
   # parse and Fix the command Line as Required by lp
   #if [ `/usr/bin/uname -r` = "5.8" ]
   #then
     #getLpCommandLine `echo $@ | tr -d "\""`
   #else
     getLpCommandLine "$@"
   #fi
   $PRNCMDPATH
  fi
  # exit with the command's exit code , This will tell the
  # server Print module if the command completed successfully
  # or not.
  exit $?
fi
done
```

5.7.2 PCL Printing Issues

Why do fields that appear as gray on my PC print as white on a UNIX PCL printer?

PCL color printing is not supported. When the pattern is set to transparent, PCL printing uses the white pen (in PCL language) to draw. When the pattern is set to a solid pattern, it uses the black pen. This behavior occurs irrespective of what color is set for the foreground or background. PostScript printing logic is different. It uses the foreground color set when the pattern is solid and the background color set when the pattern is transparent.

What PCL level is supported in Oracle Reports?

The Oracle Reports PCL driver currently supports the features of PCL Level 3. It does support HPD files for later PCL versions, but it will not honor the additional features introduced since PCL Level 3.

5.7.3 PostScript Printing Issues

What is the work around for duplex printing on PostScript printers?

You should have a printer with a duplex option and an appropriate PPD file. The example that follows was tested with a PPD file for the Kyocera FS-9000 printer. You also need the UNIX sed tool named to filter the output file.

The problem with duplexing is that it is enabled at the job level, but it gets reset in the page setup because the paper size and printer tray information are generated for every page. To work around this problem, you need a script that removes the page level setup information to avoid resetting the duplex setting. A side effect of this work around is that you cannot switch the printer tray between pages.

1. Write a sed script with the following three lines:

```
/^%%BeginPageSetup/,$ {
   /^%%BeginFeature/,/^%%EndFeature/d
}
```

2. Save the script to a file named duplexsed.

3. Copy duplexsed to an appropriate directory, such as *ORACLE_HOME*/bin.

4. Set the environment variable TK_PRINT as follows:

```
TK_PRINT="sed -f $ORACLE_HOME/bin/duplexsed | lpr -l -s -P'%n' -#'%c'"
export TK_PRINT
```

> **Note:** Print commands differ for various kinds of UNIX. Check your installation guide and man pages for your platform. Refer to Environment Variables for a description of TK_PRINT.
>
> The command stored in TK_PRINT is only executed if DESTYPE=PRINTER. If DESTYPE=FILE, you still get a PostScript file with page level setup information. You can run the duplexsed script against the PostScript file to correct it.

What PostScript level is supported in Oracle Reports?

Oracle Reports supports PostScript Level 1 and 2.

How do you dynamically change the printer tray setting in the midst of a print job?

In some cases, you may want to switch printer trays in the middle of a report. For example, you might want the first page of a report printed on letterhead stationary and subsequent pages printed on plain white paper. For character mode reports, you can achieve this result through a combination of editing the .prt file and changing the report's properties. For bit-mapped reports, you use the `SRW.SET_PRINTER_TRAY` built-in function. On UNIX, this functionality is supported for PostScript output but not PCL output. For PCL, Oracle Reports ignores the commands for changing orientation and paper tray. Although dynamically changing the orientation and printer tray for PCL is not supported on UNIX, you can change them at runtime through the print dialog box for PCL.

By using the Before Report, Between Pages, or format triggers you can switch to different printer trays as your report formats. This allows you to easily print pages of the same report on different sheets of paper.

`SRW.SET_PRINTER_TRAY` is a built in Oracle Reports function for setting the printer tray. The *Reports Builder online help* system describes this function's syntax.

Example

From the Before Report trigger, you can set the printer tray for the very first page:

```
function BeforeReport return boolean is
begin
    srw.set_printer_tray('UPPER PAPER TRAY');
    return (TRUE);
end;
```

To set the printer tray dynamically for subsequent pages, add a format Trigger to an item that prints on each page of the report. The following code checks for even pages and sets the page number accordingly:

```
function B_tbpFormatTrigger return boolean is
page_num number;
begin
   srw.get_page_num(page_num);
  begin
 if mod(page_num, 2) = 0 then
   srw.set_printer_tray('UPPER PAPER TRAY');
```

```
  else
   srw.set_printer_tray('LOWER PAPER TRAY');
 end if;
  return (true);
end;
end;
```

Why does the external print command ignore the tray select option while trying to print the PostScript output generated by Oracle Reports?

Suppose that you enter the following print command:

```
- lp -dprinter -oupper $report_print_file1
```

In this case, the -oupper option in the lp command is ignored. The reason for this behavior is that Oracle Reports generates tray information in its PostScript output. The tray selection in the PostScript overrides the specification on the command line. If you want the tray information on the command line to be respected, you need to remove the tray information from the PostScript file. You can do this by searching for and removing the following from your PostScript file:

```
%%BeginFeature: *InputSlot name of printer tray
....
%%EndFeature
```

For more information on switching printer trays, refer to How do you dynamically change the printer tray setting in the midst of a print job?

5.7.4 Font-Related Printing Issues

How do you check whether a font is used in Oracle Reports printing?

PostScript files have a list of fonts, which is created after reading the PPD file. If you examine the PostScript file, you can check the fonts by looking for the following tags:

- `DocumentNeededResource` has the list of fonts referenced in the PPD file.

- `DocumentSuppliedResource` has the list of fonts for which the PostScript driver was able to find corresponding AFM files.

- `%%Page` before the field's `%IncludeResource:font` has the font name that will be used for the field.

For PCL output files, you can check whether a particular font was used. Depending on this information, the font settings in Oracle Reports or the printer can be modified.

> **See Also:** Chapter 4, "Managing Fonts in Oracle Reports"

For more font-related information.

What is the real difference between running reports to Screen and Preview?

Formatting a report to Screen, for screen fonts, guarantees that the report will look good in the Paper Design view of the Report Editor. If an attempt is made to print a report formatted with screen fonts, though, it is likely to come out with some differences because screen fonts typically map very poorly to printer fonts. If Preview is selected instead of Screen, the report is formatted with printer fonts and the output on the screen is almost certain to match the printed output.

Will there be any font issues if I do not have a valid printer installed?

Earlier versions of OracleAS Reports Services required a valid printer on UNIX for fonts. When no valid printer was available, OracleAS Reports Services used the screen fonts, which again required the DISPLAY variable mentioned above. In OracleAS Reports Services 10*g* (9.0.4) includes a default screen printer surface, ScreenPrinter, that emulates a screen or printer for fonts in the absence of an available printer. As a result, OracleAS Reports Services does not require a printer on UNIX as of OracleAS Reports Services 10*g* (9.0.4).

> **See Also:** Chapter 5, "Printing on UNIX with Oracle Reports"

For more information on DISPLAY and printer dependencies on UNIX.

5.7.5 Printed Output Issues

Why does my report look okay on the screen but have truncated data when printed?

Any one of a number of possible causes may account for the truncation of fields.

- Check the field and determine if it is allowed to expand.

  1. In Reports Builder, double-click the field in the Paper Design or Paper Layout view to display the Property Inspector.

2. Find the Horizontal Elasticity property.

3. If it is set to Fixed, you should change it to Variable or Expand.

4. Run the report to the printer.

5. If it still truncates, it could be that the field requires multiple lines.

6. Return to the Property Inspector for the field and check its Vertical Elasticity.

7. If it is set to Fixed, you should change it to Variable or Expand.

8. Run the report to the printer again.

■   If the right most fields on the page are always the ones truncating, it could be an issue with the printable area of the printer. If you are using a PCL printer, then you will have to estimate the size of the printable area and resize your margins accordingly:

1. Open the report in Reports Builder.

2. Go to the Paper Layout view.

3. Click on the Margin tool on the top tool bar. A thick black line appears indicating where the body of your report ends and the margin begins.

4. Click and drag the black line to the left approximately 0.5 inches.

5. Save and run the report to the printer again.

6. If necessary, repeat steps 4 and 5 to determine approximately where the printable area boundary is located and then ensure that your report body fits within that area.

If you are using a PostScript printer, you can get the printable area boundary to appear in the Paper Layout view as follows:

1. Open the report in Reports Builder.

2. Choose **File > Page Setup**.

3. Verify that the margins are small and that the orientation is correct.

4. Click **OK**. The Paper Layout view should now be able to read the boundary.

5. Go to the Paper Layout view.

6. Click on the Margin tool on the top tool bar. A thick black line appears indicating where the body of your report ends and the margin begins. A

black hashed line also appears indicating the boundary of the printable area.

7. Ensure that the thick black line is inside of the black hashed line. If it is not, click and drag the black line inside the printable area.

8. Click the Margin tool to leave margin mode.

9. If necessary, reposition your fields to fit within the new body boundaries.

10. Save and run the report to the printer.

- For PCL, if it is still truncating, try using a fixed space font instead of a proportional font. Sometimes PCL printers have problems interpreting proportional space fonts and it leads to truncation. You should try using a fixed space font, such as Courier, and possibly font aliasing.

    See Also: Chapter 4, "Managing Fonts in Oracle Reports"

    For more font-related information.

    Note: Default layouts are built against a generic printer. Each printer has its own printable area. As a result, you may have to reset the report to fit the printer. Ideally, if you know the various printers you will be using, you can design the report from the start to fit the printer with the smallest printable area.

6

Using PDF in Oracle Reports

Adobe Portable Document Format (PDF) is a universal file format that preserves all the fonts, formatting, graphics, and color, of any source document regardless of the application and platform used to create it. Oracle Reports was one of the first report generation tools to embrace this technology and generate quality PDF documents.

This chapter contains the following main sections:

- PDF Features Included in Oracle Reports

  This section contains information on the various PDF features supported by Oracle Reports. This includes compression, font aliasing, font subsetting, font embedding, accessibility, and taxonomy.

- Resolving PDF Font Issues During Cross-Platform Deployment

  This section contains information on resolving PDF font issues that occur when you design a report on a Windows platform and deploy it on a UNIX platform.

- Generating a Unicode PDF File

  This section contains information on how to generate a PDF file using Unicode character sets.

- Generating a BiDi PDF File

  This section contains information on how to generate a PDF file using bidirectional (BiDi) languages such as Hebrew and Arabic.

- Generating a Multibyte PDF File

  This section contains information on how to generate a PDF file using multbyte fonts.

- Generating a Barcode PDF File

  This section contains information on how to generate a Barcode PDF file.

6.1 PDF Features Included in Oracle Reports

Oracle Reports supports PDF 1.4 and is capable of generating high fidelity PDF reports on all platforms. The PDF features supported by Oracle Reports include:

- Compression
- Font Related Features
- Accessibility
- Taxonomy
- Enhanced Graph Support

6.1.1 Compression

PDF compression decreases the PDF file size, thereby reducing the time spent in downloading the PDF file.

The amount of space saved using compression varies based on the contents of the report, for example, the number of images versus the size of the content.

- **Images**: PDF compression does not significantly affect the size of files containing images in it, as image files are typically already compressed.
- **Formatted data**: Highly formatted data can achieve higher compression rates. However, actual compression rates will vary for each report.

Compressed files are about one fifth the size of the original file. Testing has shown that the best case compression ratio of one-eigth to the worst case compression ratio of one-half was achieved based on the contents in the original file.

6.1.1.1 Setup

Oracle Reports implements a command line option to control the compression level of the PDF output. The PDF output from Oracle Reports is compressed, by default, and can be switched off or varied using the command line option: PDFCOMP.

Table 6–1 indicates which commands can use the PDFCOMP keyword.

Table 6–1   Commands that can use PDFCOMP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use PDFCOMP to specify whether PDF output should be compressed.

Syntax   PDFCOMP=*value*|{YES|NO}

Values

- *value* Any value 0 through 9. A value of 0 means the PDF file will not be compressed. A value of 1 through 9 will compress the PDF file and permit users to control the compression level.

> **Note:** Levels 0 through 9 provide incremental levels of compression. 0 being no compression and 9 being the maximum amount of compression. The greater the compression you apply, the longer it will take the report to generate the PDF file.

- YES Compresses output at compression level 6.
- NO Compresses output at compression level 0 (no compression).

Default   6

Although compressed files download quickly, the time taken to generate a compressed file is much slower when compared to a non-compressed file.

Figure 6–1   Compressed Output vs. Non-Compressed Output

> **Note:** Compression rate depends on the report's content; thus, the time taken to generate the PDF file as well as the PDF file size will vary from report to report.

6.1.2 Font Related Features

6.1.2.1 Font aliasing

Font aliasing enables you to substitute one font for another, i.e., font-to-font substitution. This font-to-font substitution is usually used when porting applications (in this case, your PDF file) across platforms. You can alias multibyte fonts as well as character sets.

Font aliasing occurs at the time of generating the PDF file. The PDF file will contain only the necessary font information required to display the output. The fonts used will not be embedded in the PDF file.

> **Note:** The fonts *must* be available on the machine *displaying* the PDF output. The fonts need not be available on the machine generating the PDF file.

At the time of viewing the report, Adobe Acrobat replaces the aliased fonts based on the following:

1. If the fonts do not exist on the machine displaying the output, Adobe Acrobat substitutes it with the `Adobe Sans MM` font.

2. If the `Adobe Sans MM` font does not match, the output may display dots for the data.

Font aliasing will work with any or all of the following:

- Single byte fonts, including Eastern European fonts for both ASCII and IS0-Latin character sets

- Adobe multibyte Character ID (CID) fonts, both multibyte and Unicode fonts, which are available as a free download from Adobe

- Type 1 Postscript fonts

- TrueType fonts

6.1.2.1.1 Setup

There are no command line options for font aliasing.

Include the font aliasing entries in the `uifont.ali` file. Oracle Reports aliases the font only when the entries in the `uifont.ali` file match the font information included in the generated PDF file.

> **Note:** The uifont.ali file is located in:
>
> - *ORACLE_HOME*\tools\common90 (Windows)
>
> - *ORACLE_HOME*/guicommon9/tk90/admin (UNIX)
>
> The uifont.ali file is the configuration file controlling all the Oracle Reports PDF font enhancements. See Managing Fonts in Oracle Reports, for more information.

The section for font aliasing in the uifont.ali file is [PDF].

The entry in the uifont.ali file for:

- Single byte fonts

```
[PDF]
"Font Name"="Font Name"
```

> **Note:** The font name entries should be enclosed within double quotes for font names containing two or more words. For example, "Brush Script MT".

- Multibyte fonts

```
[PDF]
Character Set = "Font Name"
```

            or

```
"Font Name"....Character Set="Font Name"
```

> **Note:** The font name entries should be enclosed in double quotes for font names containing two or more words. For example, "HeiseiKakuGo-W5-Acro".

Here is an example of a font aliasing entry in the uifont.ali file:

```
[ PDF ]
/*Alias TrueType to available Type 1 font */
"Kino MT" = UtopiaBold
```

```
/*Alias multibyte to available CID font */
.....SJIS = "HeiseiKakuGo-W5-Acro"
```

where:

- In `"Kino MT" = UtopiaBold`, Oracle Reports substitutes every `Kino MT` character found, with the `UtopiaBold` equivalent.

- In `.....SJIS = "HeiseiKakuGo-W5-Acro"`, Oracle Reports substitutes every multibyte character set found, with the `HeiseiKakuGo-W5-Acro` (CID) equivalent.

6.1.2.1.2 Troubleshooting

If font aliasing does not work, verify that:

- In Acrobat Reader, click **File > Document Properties > Fonts.** Verify that the aliased font has been added to the list. If it is not included, then font aliasing did not occur. The fonts were not found or the entry in the `uifont.ali` file is incorrect.

- The fonts specified for the report are available on the machine where the report will be viewed.

- The `[PDF]` section name in the `uifont.ali` file has not been modified as Oracle Reports parses the file for the section name.

- The version of the Adobe Acrobat Reader used for viewing is version 3.0 or higher. This is the version required for multibyte character reports to display properly.

6.1.2.2 Font Subsetting

PDF font subsetting is a variant of PDF font embedding. This option includes only the glyphs and character information that is actually used in the document.

With font subsetting, there is no dependency on the machine having the fonts installed. PDF font subsetting works for both single and multibyte fonts and is the preferred method of creating multibyte reports.

> **Note:** Font subsetting will work only if the glyphs and font data are included in the PDF file.

6.1.2.2.1 Setup

There are no command line options for font subsetting.

Before using the font subsetting feature you must:

- Include the font file paths in the REPORTS_PATH environment variable. Oracle Reports looks for fonts in the path specified in the REPORTS_PATH environment variable when generating a PDF file.

- Include the font subsetting entries in the uifont.ali file. Oracle Reports subsets the fonts only when the font entries listed in the uifont.ali file exist in the PDF file being generated.

The section for font subset in the uifont.ali file is [PDF:Subset].

The entry in the uifont.ali file is:

```
[PDF:Subset]
Font Name = "Font File Name"
```

The file name entries should be enclosed within quotes; however, only font names containing two or more words should be enclosed within quotes.

The font file name referenced in the uifont.ali file must match the existing font file name for font subsetting to work correctly.

> **Note:** The font file name is not the font name displayed in Reports Builder.

Example

```
[PDF:Subset]
Arial = "Arial.ttf"
```

> **Note:** Oracle Reports supports TTF as well as TTC.

When you subset a font in a PDF file, it becomes a custom font because it contains only those characters needed for the report output.

Using Adobe Acrobat 3.0 or higher, you can view the fonts used in your report:

- **File > Document Properties > Fonts**.

■ The Document Font dialog box displays Original Font, Type, Encoding, Actual Font (or the font used), and Type.

Figure 6–2   Font Subsetting
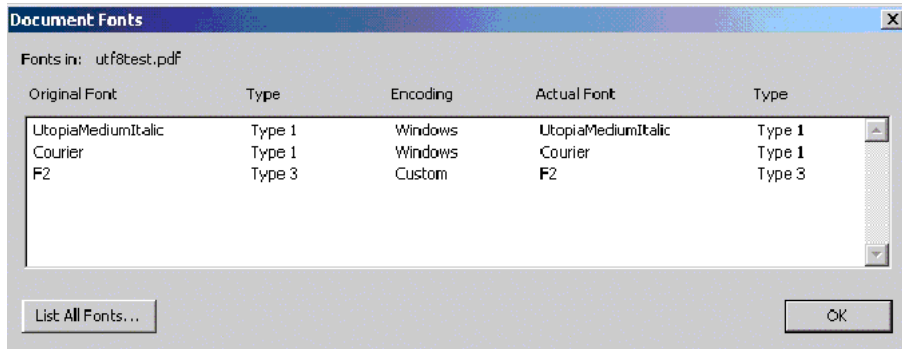
> **Note:**   In the case of font subsetting:
>
> ■ The **Encoding** column will display **Custom**
>
> ■ The **Actual Font** column will display **F2**
>
> ■ The **Type** column will display **Type3**
>
> Type3 fonts are imaged characters that look slightly bolder than they would if expressed as a Type1 font.

Recommended

For optimum viewing, we recommend that you use Adobe Acrobat Reader 5.0 and set the following parameters in Adobe Acrobat Reader:

1. **Edit > Preferences > General.**

2. **Display > Smoothing.**

3. Select **Smooth Text, Smooth Line Art, and Smooth Images**.

4. (Laptop/LCD Screens) Select the **Use CoolType** check box.

5. Click **OK.**

PDF font subsetting in Oracle Reports works with TrueType fonts only and not with Type1 fonts. To include specific TrueType fonts in your report, convert Type1 fonts to TrueType fonts using available 3rd party tools.

There is a limitation on UNIX platforms when working with TrueType fonts. To work around this issue do the following:

1. Copy the TrueType fonts to the UNIX machine.

2. Ensure that the correct font path is specified in the REPORTS_PATH environment variable.

3. Run the report with the associated TrueType fonts. Refer to Designing the Report on Windows and Deploying it on UNIX, for more information on running a report on UNIX machines.

6.1.2.2.2 Troubleshooting If font subsetting does not work, verify the following:

- In Acrobat Reader, click **File > Document Properties > Fonts.** Verify that the Actual Font value is Custom and Type is Type3. If this is not specified then font subsetting did not occur. The problem could be either that the fonts were not found or the entry in the uifont.ali file is incorrect.

- The font file names are valid.

- The font types are TrueType, i.e., *filename*.ttf/*filename*.ttc.

- The font name is enclosed in double quotes if it consists of two or more words.

- The font name does not contain embedded parenthesis.

- The font files are located in the path specified by the REPORTS_PATH environment variable. When generating a PDF file, Oracle Reports looks for fonts in the path specified in the REPORTS_PATH environment variable.

- The font names are correct and are available on the machine where the PDF file is generated.

- The [PDF:Subset] section name in the uifont.ali file has not been modified. Oracle Reports parses the file looking for the section name.

- The version of the Adobe Acrobat Reader used for viewing is version 3.0 or higher. This is the version required for multibyte character reports to display properly.

- Fonts that are subset look bold. This is an issue with Type3 fonts, which are created during font subsetting. Type 3 fonts are imaged characters that look slightly bolder than they would if expressed as a Type 1 font. See Recommended for more information on improving the viewing quality.

  Refer to Designing the Report on Windows and Deploying it on UNIX, for more information on running a report on UNIX machines.

6.1.2.3 Font Embedding

PDF font embedding is the process of including the entire font set along with the data in the PDF file. PDF font subsetting and font embedding are mutually exclusive.

> **Note:** Font embedding increases your PDF file size.

PDF font embedding in Oracle Reports is for Type1 fonts only (either single or multibyte fonts) and not for TrueType fonts. Convert TrueType fonts to Type1 fonts using available 3rd party tools in order to include specific Type1 fonts in your report.

PDF font embedding with Oracle Reports occurs between a font and a set of font file names.

> **Note:** You must ensure that you have the necessary font licenses before embedding any fonts in your output.

6.1.2.3.1 Setup

The setup for PDF embedding includes:

- A command line option: PDFEMBED

- An entry in the uifont.ali File Entry, [PDF:Embed]

The command line option PDFEMBED is used to specify whether Oracle Reports will embed the Type1 postscript fonts specified in the `uifont.ali` file into the PDF output.

> **Note:** Font embedding will work only if the fonts are included in the PDF file.

PDFEMBED

Table 6–2 indicates which commands can use the PDFEMBED keyword.

Table 6–2   Commands that can use PDFEMBED

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use PDFEMBED to specify whether Oracle Reports will embed the Type1 PostScript font file(s) specified in the uifont.ali file into PDF output.

Syntax   PDFEMBED={YES|NO}

Values

- YES The PDF driver will embed the font(s) specified in the [PDF:Embed] header of the uifont.ali file into the PDF output.

- NO The font(s) will not be added to PDF output.

Default   YES

uifont.ali File Entry

The section for font aliasing in the uifont.ali file is [PDF:Embed].

(Windows only) The entry in the uifont.ali file should be:

Font Name = "Font Name.pfm Font Name.pfb"

(UNIX only) The entry in the uifont.ali file should be:

Font Name = "Font Name.afm Font Name.pfa"

Example 6–1   Font Embedding

```
[PDF:Embed]
Symbol = "Symbol.pfm Symbol.pfb"
```

In Example 6–1, the Symbol font is embedded into the PDF file. This ensures portability by:

1. Creating the report with the Symbol font.

2. Embedding the Symbol font in the PDF file (Figure 6–3).

Figure 6–3   Font Embedding
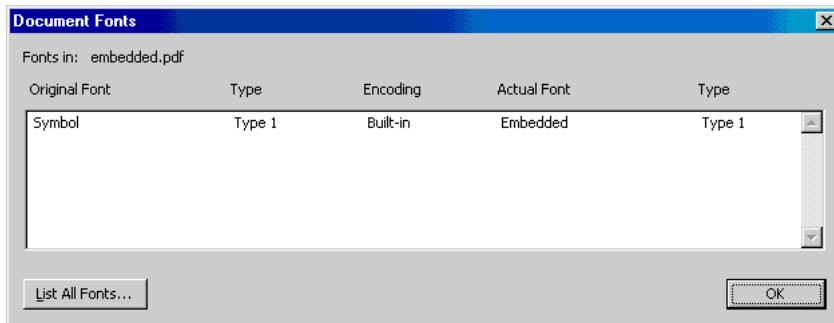
6.1.2.3.2   Troubleshooting

If PDF font embedding does not work, verify the following:

- In Acrobat Reader, click **File > Document Properties > Fonts.** Verify that the embedded font has been added to the list. If the font has not been added, then font embedding did not occur. The problem could be either that the fonts were not found or the entry in the `uifont.ali` file is incorrect.

- The correct font file name is used.

- The font path specified in the REPORTS_PATH environment variable is correct. When generating the PDF file, Oracle Reports looks for fonts in the paths specified in the REPORTS_PATH environment variable.

- The font type is a Type1 font.

- The font name is enclosed within double quotes if it consists of 2 or more words.

- The [PDF:Embed] section name in the `uifont.ali` file has not been modified. Oracle Reports parses the file looking for the section name.

- The format to specify the embedded font is valid

  ```
  Font Name="fontfilename.pfm/.afm file fontfilename.pfb/.pfa file".
  ```

  For example (Windows):

  ```
  UtopiaMediumItalic = "UtopiaMediumItalic.pfm UtopiaMediumItalic.pfb"
  ```

- The font name is correct and available on the machine where the PDF file is generated.

6.1.3 Precedence of Execution

The precedence order for the same font in multiple places within the `uifont.ali` file is as follows:

1. Font aliasing takes precedence over font embedding (highest)

2. Font embedding takes precedence over font subsetting (intermediate)

3. Font subsetting takes no precedence (lowest)

For example, if you have included the same font entries for both font embedding and font subsetting, then font embedding will override font subsetting. This is assuming you have not set the command line option of `PDFEMBED=NO`.

For all font features —font aliasing, font subsetting, and font embedding—include the specific entries first followed by the generic entries. For example, if you want to subset Arial Plain, Arial Bold, Arial Italic, and Arial Bold-Italic fonts, your entries should be in the following order:

```
[ PDF:Subset ]
Arial..Italic.Bold.. = "Arialbi.ttf"
Arial...Bold.. = "Arialb.ttf"
Arial..Italic... = "Ariali.ttf"
Arial..... = "Arial.ttf"
```

If the plain `Arial..... = "Arial.ttf"` entry appears first, then all the styles of the Arial font in the layout will be subset as Arial Plain font. Here is a sample of a portion of the `uifont.ali` file for all the PDF entries containing all three PDF sections:

Sample 1

```
[ PDF ]
Palatino = "Kino MT.ttf"
[ PDF:Subset ]
Garmond..Italic.Bold.. =
"Garmacbi.ttf"
Garmond...Bold.. = "Garmacb.ttf"
Garmond..Italic... = "Garmaci.ttf"
Garmond..... = "Garamac.ttf"
[ PDF:Embed ]
Arial = "Arial.pfm Arial.pfb"
```

Sample 2

```
[PDF]
```

```
Arial.10.Italic = "Times New Roman".12.Italic.Bold
"Courier New" = Symbol
[PDF:Embed]
"Times New Roman".14..Bold = "TimesBold.pfm TimesBold.pfb"
[PDF:Subset]
Verdana..Italic.Bold = "Verdanaz.ttf"
Verdana…Bold = "Verdanab.ttf"
```

Table 6–3   Value Comparison of the various PDF types

PDF Type	Advantage	Disadvantage
Font Aliasing	Multibyte support Good display Small file size (Japanese example; 23kb for font aliasing when compared to 130kb for font subsetting)	Unicode character set not supported Asian Font Packs are required on the client machine, if the client's operating system and Acrobat Reader are not the native version. Limited fonts support . For example, there is no support for font emphasis.
Font Embedding	Guaranteed display	Only single byte support provided. Large file size.
Font Subsetting	Unicode support Guaranteed display	Inadequate display (*) No styles (Italic and Bold) support The generated PDF file cannot be edited using Acrobat Reader.

(*) Refer to Recommended for more information on how to correct this.

6.1.4 Accessibility

Oracle Reports provides several ways for you to include accessibility features in your PDF file. The PDF format file follows the tagged-PDF standard defined in PDF 1.4. This standard along with Acrobat Reader 5 (or higher) provides you with features for inclusion in the paper layout. The command line option to include accessibility features in your output is ACCESSIBLE.

Table 6–4 indicates which commands can use the ACCESSIBLE keyword.

Table 6–4   Commands that can use ACCESSIBLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use ACCESSIBLE to specify whether accessibility-related features offered through Oracle Reports are enabled (YES) or disabled (NO) for the PDF file.

Syntax   ACCESSIBLE={YES|NO}

Values

- YES   Accessibility features are enabled for PDF file.

- NO   Accessibility features are not enabled for PDF file.

Default   NO

Using the additional properties (Figure 6–4), you can provide the required information necessary for generating PDF documents that can be interpreted by assistive technology. For more information on accessibility, see the *Creating Accessible Enterprise Reports using Oracle9i Reports* white paper on (http://otn.oracle.com)under Accessibility.

Figure 6–4   Accessibility
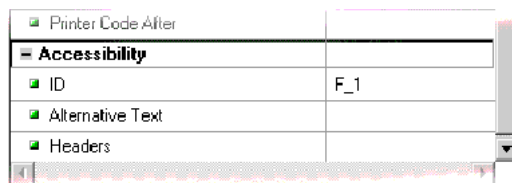
6.1.5 Taxonomy

A PDF document can include global information about itself such as the document's title, author, creation and modification dates. This global information proves useful at the time of cataloguing or searching for documents in external databases.

Oracle Reports provides properties to enable such a classification, otherwise known as taxonomy. They are:

- Title

- Author

- Subject

- Keywords

Table 6–5   Taxonomy Properties

Property Name	Type	Description	Default Value
Title	String	Document title	PDF document name
Author	String	Document's author	Oracle Reports
Subject	String	Document's subject	None
Keywords	String	Specifies keywords that can be used to categorize the document	None

> **Note:**   The taxonomy properties are report level properties.

Refer to the *Reports Builder online help* for more information on the taxonomy properties.

6.1.6 Enhanced Graph Support

Oracle Reports now provides enhanced support for graphs in PDF. This includes specifying a higher dots per inch (DPI) value to increase or improve the image resolution of the graph. The improved graph support enables you to scale the graph without compromising on the image quality.

The environment variable is REPORTS_GRAPH_IMAGE_DPI.

Description   The REPORTS_GRAPH_IMAGE_DPI environment variable specifies a dots per inch (DPI) value for graphs being output to a printer (PDF, PostScript, or PCL). This environment variable allows you to increase the earlier fixed default value of 72 DPI, which helps in improving the image resolution for printed (PDF, PostScript, or PCL) graphs.

Valid Values   72 through 300

Default   250

Usage Notes
- On Windows, use the registry to specify the value. On Unix/Linux, set the environment variable in reports.sh.

- Generating a high resolution chart will increase the PDF file size relatively and the time taken to generate the PDF file/printer output.

- When you set a DPI value greater than 250 and your chart is bigger than 5"x5" (approximately), you may also need to change the JVM heap size value via REPORTS_JVM_OPTIONS to avoid the Out Of Memory error for the JVM. See REPORTS_JVM_OPTIONS, for more information on setting the JVM options

- To revert to the old behavior, set the value of this environment variable to 72 DPI.

- This variable is currently not supported in Oracle Reports distribution functionality as this is specific to PDF, PS, and printer outputs only.

6.2 Resolving PDF Font Issues During Cross-Platform Deployment

There are font and text alignment issues when you design a report (single byte or multibyte) on the Windows platform and deploy it on a UNIX platform. The reason is that the font handling and windowing system are completely different across the two platforms.

6.2.1 Designing and Deploying the Report on the Same Platform

Your report is designed and deployed on the same platform, e.g., Windows:

- There should be no font or text alignment issues in the PDF file.

- If the PDF file is generated with font subsetting enabled, then the PDF file can be viewed in the same manner across platforms.

6.2.1.1 Designing the Report on Windows and Deploying it on UNIX

Your report is designed on the Windows platform and deployed on the UNIX platform.

(Windows) You use the TrueType fonts located in the Windows machine. Oracle Reports queries the font information from the Windows system for formatting the report.

(UNIX) When this report is sent to PDF on Solaris or any other UNIX platforms, there are two stages:

1. Oracle Reports renders the font metrics information for the fonts and uses this information to format various objects in the report.

   > **Note:** Oracle Reports renders the font metrics information from the AFM files mentioned in the printer's PPD file.

2. Oracle Reports then looks for the entries in the [PDF] section of the uifont.ali file. For font subsetting, Oracle Reports refers to the [PDF: Subset] section and subsets the TrueType fonts from the given location. The subsetted fonts are then embedded in the PDF file.

   > **Note:** The corresponding AFM files for all the TrueType fonts used in your report should be available on the UNIX machine to ensure adequate formatting is enforced.

6.2.1.1.1 Step by Step Procedure for Single Byte fonts This section outlines the steps involved in generating a PDF report (using single byte fonts) designed on the Windows platform on either Solaris or any other UNIX platform. These steps are required only if you see font alignment issues in your PDF output.

1. Create a report on the Windows platform with TrueType fonts. For this procedure, the fonts referred to are arial.ttf and tahoma.ttf.

2. Copy the fonts (arial.ttf and tahoma.ttf) and your report's .rdf file to the UNIX platform. The path for the font files should be *ORACLE_HOME/font folder*. Add the font file's path to the REPORTS_PATH environment variable.

3. Create the AFM files for the font files (arial.ttf and tahoma.ttf).

4. Copy the AFM files (arial.afm and tahoma.afm) generated to *ORACLE_HOME*/guicommon9/tk90/admin/AFM.

> **Note:** The AFM files should be copied to the AFM directory without the `.afm` extension. Additionally, ensure that the name of `afm` file, the name of the font in the .ppd file, and the name of the font the `uifont.ali` file are an exact match.

5. Ensure that the `TK_PRINTER` environment variable or the `PRINTER` environment variable is set to the printer name. For example, `hrprinter`.

6. Ensure that the `uiprint.txt` file has the following entry:

   printer name`:PostScript:2:test:default.ppd:`

   For example:

   `hrprinter:PostScript:2:test:default.ppd:`

7. Add the AFM entries to the `ppd` file.

   > **Note:** This `ppd` file is the first entry in the `uiprint.txt` file and contains your font information. The default `ppd` file is `datap462.ppd.`

   ```
   *Font arial: Standard "(001.001)" Standard ROM
   *Font tahoma: Standard "(001.001)" Standard ROM
   ```

8. Ensure that there are no entries in the [`PDF:Subset`] section at this time in the `uifont.ali` file.

9. Run the report to generate the PDF file. In Acrobat Reader, click **File > Document Properties > Fonts** or **File >Document Info > Fonts**:

   a. The **Original Font** column displays the Arial and Tahoma fonts.

   b. There will be some font alignment issues.

10. Add the following entry in the `uifont.ali` file:

    ```
    [ PDF:Subset ]
    "arial" = "arial.ttf"
    "tahoma" = "tahoma.ttf"
    ```

11. Run the report again to generate the PDF file. The PDF file should not contain any font alignment issues.

To confirm that the fonts are subset in the PDF file:

a. In Acrobat Reader, click **File > Document Properties > Fonts.**

b. The **Original Font** column should display **F2**, the **Encoding** column should display **Custom**, and the **Type** column should display **Type3**.

6.2.1.1.2 Step by Step Procedure for MultiByte and Unicode fonts There are additional steps for generating reports with multibyte fonts. The steps involved in resolving font issues with PDF subsetting when deploying multibyte reports on UNIX platforms are as follows:

1. Create a report on the Windows platform using TrueType multibyte fonts with the appropriate character set. For this procedure the font and the character sets referred to are the Korean font `h2mjsm.ttf` and the `KO16KSC5601` character set.

2. Copy the Korean font `h2mjsm.ttf` and your report's `.rdf` file to the UNIX platform. The font file path should be `$ORACLE_HOME/`*font folder*. Add the font file's path to the `REPORTS_PATH` environment variable.

3. Create the AFM files for the Korean font `h2mjsm.ttf`.

4. Copy the AFM file to the following location:

```
$ORACLE_HOME/guicommon9/tk90/admin/AFM/.
% cp h2mjsm.afm ORACLE_HOME/guicommon9/tk90/admin/AFM/h2mjsm
```

5. Ensure that the `TK_PRINTER` environment variable or the `PRINTER` environment variable is set to the printer name. For example, `hrprinter`.

6. Ensure the `uiprint.txt` file has the following entry:

```
printer name:PostScript:2:test:default.ppd:
```

For example:

```
hrprinter:PostScript:2:test:default.ppd:
```

7. Add the following lines in the `.ppd` file:

Note: This `ppd` file is the first entry in the `uiprint.txt` file and contains your font information. The default `ppd` file is `datap462.ppd`.

```
*DefaultFont: h2mjsm
```

```
*Font h2mjsm: Special "(001.001)" Special ROM
```

8. Comment the Symbol line in the file:

```
*%Font Symbol: Special "(001.001)" Special ROM
```

9. Edit the following section in the `uifont.ali` file to mention the font used for the character set:

> **Note:** The `uifont.ali` file is located in:
>
> - *ORACLE_HOME*\tools\common90 (Windows)
>
> - *ORACLE_HOME*/guicommon9/tk90/admin (UNIX)
>
> The `uifont.ali` file is the configuration file controlling all the Oracle Reports PDF font enhancements. Refer to Managing Fonts in Oracle Reports, for more information.

```
[ Global ]
.....ko16ksc5601 ="h2mjsm"
[ Printer:PostScript2 ]
.....ko16ksc5601 ="h2mjsm"
```

10. Ensure that there are no entries in any of the [PDF] or [PDF:Subset] sections at this time in the `uifont.ali` file.

11. Run the report to generate the PDF file. In Acrobat Reader, click **File > Document Properties > Fonts** or **File > Document Info > Fonts**:

    a. The **Original Font** column displays the **h2mjsm** font.

    b. There will be some font alignment issues.

12. Add the following entries in the `uifont.ali` file to enable PDF subsetting:

```
[ PDF:Subset ]
"h2mjsm"="h2mjsm.ttf"
```

13. Run the report again to generate the PDF file. The PDF file should not contain any font alignment issues.

    To confirm that the fonts are subset in the PDF file:

    a. In Acrobat Reader, click **File > Document Properties > Fonts** or **File > Document Info > Fonts.**

   b. The **Original Font** column should display **F2**, the **Encoding** column should
   display **Custom**, and the **Type** column should display **Type3**.

> **Note:** There might be some variations in the alignment, as the font
> metrics handling is different in UNIX and Windows. This issue
> cannot be avoided.
>
> A PDF file generated with the font subsetting enabled might have
> some font style issues (e.g., some content could be displayed as
> bold) when viewed in Acrobat Reader. This is because Acrobat
> Reader has a limitation while displaying the Type 3 fonts. See
> Recommended for more information on how to smoothen the
> display for Type 3 fonts.

6.3 Generating a Unicode PDF File

This section outlines the steps involved in generating a PDF file with a Unicode
character set.

6.3.1 Font Subsetting

The steps involved in generating a Unicode PDF file using the font subsetting
feature are as follows:

1. Set NLS_LANG=AMERICAN_AMERICA.UTF8.

2. Set REPORTS_PATH to the font directory in which the TrueType font exists. For
   example, C:\WINNT\Fonts.

3. Open the uifont.ali file and edit the [PDF:Subset] section to specify the
   TrueType font name.

> **Note:** The uifont.ali file is located in:
>
> - *ORACLE_HOME*\tools\common90 (Windows)
>
> - *ORACLE_HOME*/guicommon9/tk90/admin (UNIX)

Example

```
[ PDF:Subset ]
"Andale Duospace WT J" = "Aduoj.ttf"
"Albany WT J"="AlbanWTJ.ttf"
```

The specified font should cover the Unicode range that your report uses.

4. Create a report having MLS data and set its font to the Unicode font.

5. Run a report having MLS data with DESTYPE=FILE DESFORMAT=PDF.

6.4 Generating a BiDi PDF File

This section outlines the steps involved in generating a PDF file for bidirectional (BiDi) languages.

Oracle Reports provides two environment variables that resolve font re-shaping and numeric options with bidirectional (BiDi) languages, such as Hebrew and Arabic. They are:

1. REPORTS_BIDI_ALGORITHM

   This environment variable switches the layout algorithm for bidirectional (BiDi) languages (for example, Arabic or Hebrew). The valid values for this environment variable are ORACLE or UNICODE.

   See Also: REPORTS_BIDI_ALGORITHM

   For more information on the environment variable.

2. REPORTS_ARABIC_NUMERAL

   This environment variable specifies the numeric format for Arabic PDF output.

   See Also: REPORTS_ARABIC_NUMERAL

   For more information on the environment variable.

6.4.1 Font Subsetting

The steps involved in generating a PDF file for bidirectional (BiDi) languages using the font subsetting feature are as follows:

1. Set NLS_LANG=ARABIC_EGYPT.AR8MSWIN1256 (or AR8ISO8859P6 on UNIX)

2. Set REPORTS_PATH to the font directory in which the TrueType font exists. For example, C:\WINNT\Fonts.

3. Open the uifont.ali file and edit the [PDF:Subset] section to specify the TrueType font name.

> **Note:** The `uifont.ali` file is located in:
>
> - *ORACLE_HOME*\tools\common90 (Windows)
> - *ORACLE_HOME*/guicommon9/tk90/admin (UNIX)

Example

```
[PDF:Subset]
"Andale Duospace WT J" = "Aduoj.ttf"
"Albany WT J"="AlbanWTJ.ttf"
```

4. Create a report having Arabic data and set it to the font specified in the example.

5. Run a report with `DESTYPE=FILE DESFORMAT=PDF`.

6.5 Generating a Multibyte PDF File

This section outlines the steps involved in generating a PDF file with multibyte fonts.

Figure 6–5 outlines the mapping table between Oracle `NLS_CHARACTERSET`, CMap name, and its CID font name used in aliasing pdf.

Figure 6–5   CID Font Mapping for Aliasing PDF

Language	Oracle NLS_CHARACTERSET name	CMap name	CIDFont name
Japanese	JA16SJIS	90ms-RKSJ-H	"KozMinPro-Regular-Acro" (*) "HeiseiKakuGo-W5-Acro" (**) "HeiseiMin-W3-Acro" (**)
	JA16EUC	EUC-H	
Korean	KO16KSC5601	KSC-EUC-H	"HYSMyeongJoStd-Medium-Acro" (*) "HYGoThic-Medium-Acro" (**) "HYSMyeongJo-Medium-Acro" (**)
	KO16MSWIN949	KSCms-UHC-H	
Traditional Chinese	ZHT32EUC	CNS-EUC-H	"MSungStd-Light-Acro" (*) "MHei-Medium-Acro" (**) "MSung-Light-Acro" (**)
	ZHT16BIG5, ZHT16MSWIN950	ETen-B5-H	
	ZHT16HKSCS	HKscs-B5-H	"MSungStd-Light-Acro"(*)
Simplified Chinese	ZHS16CGB231280	GB-EUC-H	"STSongStd-Light-Acro" (*) "STSong-Light-Acro"(**)
	ZHS16GBK	GBK-EUC-H	

(*) Those fonts are available Adobe in Acrobat Reader Version 5

(**) Those fonts are available Adobe in Acrobat Reader Version 4

It is recommended that you use Version 5 CIDFonts(*) with Acrobat Reader 5.0 in order to avoid unexpected font mapping, which results in multibyte characters overlapping.

6.5.1 Font Aliasing

The steps involved in generating a PDF file for multibyte fonts using the font aliasing feature are as follows:

1. Set `NLS_LANG=JAPANESE_JAPAN.JA16SJIS` (or `JA16EUC` on UNIX)

2. Open the `uifont.ali` file located and set the font alias under the `[PDF]` section.

> **Note:** The `uifont.ali` file is located in:
>
> - *ORACLE_HOME*\tools\common90 (Windows)
> - *ORACLE_HOME*/guicommon9/tk90/admin (UNIX)

Example

```
[ PDF ]
.....JA16SJIS = "KozMinPro-Regular-Acro"
"MS UI Gothic".....JA16SJIS = "KozMinPro-Regular-Acro"
```

3. Create a report having Japanese data with the Japanese font (`MS UI Gothic`).

4. Run a report with `DESTYPE=FILE DESFORMAT=PDF`.

5. If your Acrobat Reader is a non-Japanese version installed on a non-Japanese operating system, you need to install the Japanese font pack from Adobe's site.

   If you view the PDF file with the Japanese version of Acrobat Reader 4.0/5.0 on the Japanese version of Windows, you do not need to install the Japanese font pack.

6.5.1.1 Font Subsetting

The steps involved in generating a PDF file for multibyte fonts using the font subsetting feature are as follows:

1. Set `NLS_LANG=JAPANESE_JAPAN.JA16SJIS` (or `JA16EUC` on UNIX)

2. Set the `REPORTS_PATH` environment to the font directory in which the TrueType font exists. For example, `C:\WINNT\Fonts`.

3. Open the `uifont.ali` file located in the *ORACLE_HOME*`\tools\common90` directory and edit it at the `[PDF:Subset]` section to specify the TrueType font name.

Example

```
[ PDF:Subset ]
"Andale Duospace WT J" = "Aduoj.ttf"
"Albany WT J"="AlbanWTJ.ttf"
"MS UI Gothic" = "msgothic.ttc"
```

4. Create a report having Japanese data and set it to the font mentioned in the example.

5. Run a report with `DESTYPE=FILE DESFORMAT=PDF`.

6.6 Generating a Barcode PDF File

This section outlines the steps involved in generating a PDF file with barcode information.

6.6.1 Font Embedding

The steps involved in generating a barcode PDF file using the font embedding feature are as follows:

1. Set the `REPORTS_PATH` environment variable to the font directory containing the Type1 font.

2. Open the `uifont.ali` file and include the following under the font embed `[PDF:Embed]` section.

> **Note:** The `uifont.ali` file is located in:
>
> - *ORACLE_HOME*`\tools\common90` (Windows)
> - *ORACLE_HOME*`/guicommon9/tk90/admin` (UNIX)

Example

```
[ PDF:Embed ]
SAdHC39a = "SAdHC39a.pfm SAdHC39a.pfb"
```

3. Create a report having Barcode data and set its font to the one mentioned in the example.

4. Run a report with DESTYPE=FILE DESFORMAT=PDF.

6.6.1.1 Font Subsetting

The steps involved in generating a barcode PDF file using the font subsetting feature are as follows:

1. Set the REPORTS_PATH environment variable to the directory containing the TrueType font. For example, C:\WINNT\Fonts.

2. Open the uifont.ali file in the *ORACLE_HOME*\tools\common90 directory and include the following under the [PDF:Subset] section to specify the TrueType font name:

Example

```
[ PDF:Subset ]
SAdHC39a = "SAdHC39a.ttf"
```

3. Create a report having Barcode data and set it to the font mentioned in the example.

4. Run a report with DESTYPE=FILE DESFORMAT=PDF.

7

Configuring Destinations for OracleAS Reports Services

Two things to consider when you run a report are how the report should be output (destination) and who should receive it (distribution). Distribution is discussed in Chapter 15, "Creating Advanced Distributions". This chapter explores how OracleAS Reports Services handles output processing to default and custom destinations. It provides an overview of output processing and information on registering destination types with the OracleAS Reports Services.

It includes the following sections:

- Overview of Output Processing
- Registering Destination Types with the Server

7.1 Overview of Output Processing

How the report should be output is controlled by the destype that you specify at runtime, which, in turn, are determined by the destination output types you have registered in your server configuration file (*server_name*.conf). You can register no output types and simply use the default types provided by OracleAS Reports Services:

- Cache (i.e., browser)
- SMTP-compliant e-mail
- File
- Printer
- OracleAS Portal (this is an exception in that, for access to the portal, it requires the specification of a userid and password in the server configuration file)

You can also define custom output types, such as fax, Oracle's Internet File System (*i*FS), or any type you define using the OracleAS Reports Services Destinations API. This API enables you to define new destination types and build handlers to usher your reports to custom destinations.

> **Note:** Build a custom destination type via the OracleAS Reports Services Destinations API. Look for upcoming information about Oracle Reports APIs and destination types for download on the Oracle Technology Network, (`http://otn.oracle.com`).

The OracleAS Reports Services architecture standardizes the way output is generated and delivered. It takes responsibility for delivering report output to the appropriate destination (via the Reports Server), yet generates output independent of its destination (via the Reports Engine). This provides a significant improvement in efficiency by allowing one run of a report to be used in a number of different ways. It also opens up the output processing architecture to allow for any number of destination types.

In the past, the Reports Runtime engine was totally responsible for delivering the output. Consequently, it had to know how to communicate with output destinations. This resulted in a tight coupling between the engine and the supported destinations.

OracleAS Reports Services eliminates this tight coupling and its attendant restrictions. The runtime engine now treats all destinations alike. It doesn't need to know the destination type for which the output is being produced. The server hands output off to destination handlers that prepare the material for delivery to their associated destination types. You can use predefined destination types (with predefined handlers) or create a handler for a custom destination type you intend to support. Almost any type of destination can be plugged into Oracle Reports.

Figure 7–1 illustrates the main components of the OracleAS Reports Services output processing architecture.

Figure 7–1   Main components of destination/distribution architecture
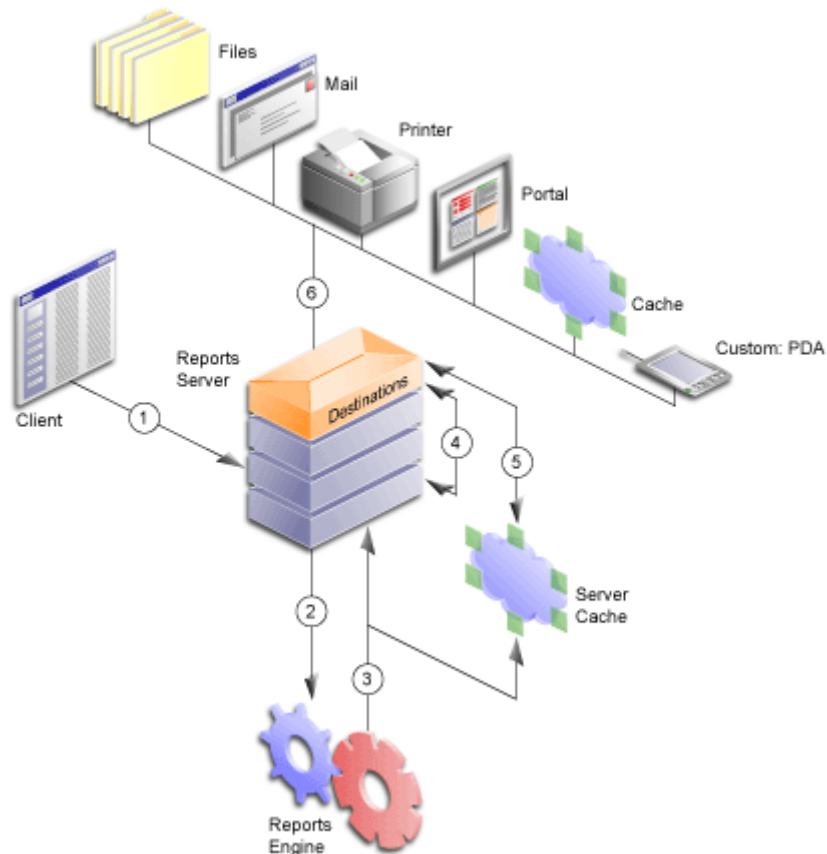
Requests flow through the output processing architecture in the following sequence:

1. The user submits a request from a client or browser to the Reports Server.

2. The server passes it along to the runtime engine.

3. The runtime engine creates/processes the destination objects (which include file lists for specific destinations as well as any properties related to those destinations) and the report output; the runtime engine sends the destination objects to the Reports Server and the report output to cache.

4. The Reports Server sends the destination objects to the Reports Server's destination component.

5. The destination component of the Reports Server fetches the report output from cache.

6. The Reports Server destination component sends the report and the destination objects (which specify how the destination device should handle the output) to the appropriate destination handler.

7.2 Registering Destination Types with the Server

Before OracleAS Reports Services can send a report to a particular destination type, the type must be a default type (printer, e-mail, cache, or file) or a type registered in the server's configuration file, *server_name*.conf. The configuration file contains a `destination` element for registering destination types that are valid for your reports. You can register anywhere from zero to any number of destination types.

Registering a destination type with the server involves:

- Setting Up a Destination Section in the Server Configuration File

- Entering Valid Values for a Destination

These tasks are described in the following sections.

7.2.1 Setting Up a Destination Section in the Server Configuration File

To set up a destination section in the *server_name*.conf file:

1. Open the server configuration file with your preferred text editor.

   You'll find the server configuration file in the following directory (Windows and UNIX use the same path):

   `ORACLE_HOME\reports\conf\server_name.conf`

2. If the configuration file does not have a `destination` section, create one underneath the element that precedes it in the configuration file's data type definition file (`rwserverconf.dtd`) section.

   Note: The server configuration file follows the order of elements defined in the file's related document type definition file (*ORACLE_HOME*\reports\dtd\rwserverconf.dtd). Place `destination` after the elements that precede it, whichever are present in your server configuration file.

   ---

3. Use the following syntax to register all the destination types you will use for outputting reports:

```
<destination destype="output_type_1" class="java_class_1">
    <property name="valid_destype_property" value="valid_value"/>
    <property name="valid_destype_property" value="valid_value"/>
</destination>
<destination destype="output_type_2" class="java_class_2">
    <property name="valid_destype_property" value="valid_value"/>
</destination>
```

The valid values for these tags are discussed in the following sections.

7.2.2 Entering Valid Values for a Destination

7.2.2.1 Destination destypes and classes

The `destype` and `class` attributes are required for valid registration of a non-default output type. They specify the destination types and their associated Java classes. The predefined (default) destination types and classes that come with OracleAS Reports Services are listed in Table 7–1:

Table 7–1   Standard destination types and classes

Destination	destype	class
OracleAS Portal content area	oraclePortal	oracle.reports.server.DesOraclePortal
SMTP-compliant e-mail	mail	oracle.reports.server.DesMail
file	file	oracle.reports.server.DesFile
cache	cache	oracle.reports.server.DesCache
printer	printer	oracle.reports.server.DesPrint
FTP	ftp	oracle.reports.plugin.destination.ftp.DesFTP
WebDAV	WebDAV	oracle.reports.plugin.destination.webdav.DesWebDAV

Contrary to the other default types, you must register an `oraclePortal` destype. This is because the `oraclePortal` destype requires the specification of a userid and password for accessing the portal.

> **See Also:**   Chapter A, "Command Line Options"
>
> For examples on pushing a report using the `oraclePortal` destype.

You are not limited to the predefined destypes and classes provided with the server. You can register custom destination types, such as a fax, once you have defined a custom handler (through the Destinations API).

> **Note:**   Look for upcoming information about Oracle Reports APIs on the Oracle Technology Network, (`http://otn.oracle.com`).

7.2.2.2 Destination Property name/value Pairs

The server configuration file allows the association of an unlimited number of properties with a registered destination. Destination properties consist of name/value pairs that define some aspect of an output type's configuration. They are expressed in terminology recognized by the destination type. For example, a destination with a destype of `oraclePortal` would recognize the name/value pair:

```
<property name="portalUserid" value="portal_id/portal_password@portal_schema"
confidential="yes" encrypted="no"/>
```

This example defines the values to be associated with a portal user ID. It includes the attributes `confidential` and `encrypted:confidential="yes"`, which indicate that the values within this element should be encrypted; `encrypted="no"`, which indicates that the values are not yet encrypted. The next time the Reports Server starts, it will automatically encrypt the values and reset `encrypted` to `yes`.

> **Note:**   Elements and attributes allowable in server configuration file are determined by the syntax defined in the `rwserverconf.dtd` file (`ORACLE_HOME\reports\dtd\rwserverconf.dtd`). This is discussed in detail in Chapter 3, "Configuring OracleAS Reports Services".

What is valid for a destination type's properties depends entirely on the destination type. These values do not come from Oracle Reports and are not put to use by the Reports Server. They come from the destination type itself and use terms the

destination recognizes. It is up to the developer to understand the requirements of a custom destination and to know what properties to associate with a given custom output type.

When we begin to discuss distribution, you may note that within the distribution XML file, the destype element also allows for the use of property name/value pairs. It's important to make a distinction between properties entered for a destination element in the server configuration file and those entered for a destype element in the distribution XML file:

- Properties entered for a destination element in the server configuration file should deal only with configuring an output type, for example setting an allowable number of retries for a destination fax.

- Properties entered for a destype element in the distribution XML file should deal only with specifying a runtime parameter, for example the identity of the fax's intended recipient.

7.2.3 Example Destination

The following example illustrate a destination element for pushing content into OracleAS Portal:

```
<destination destype="oraclePortal" class="oracle.reports.server.DesOraclePortal">
  <property name="portalUserid" value="<the_username_password_tnsname_for_logon_to_portal>"
     encrypted="yes"/>
</destination>
```

8

Configuring and Using the JDBC PDS

The JDBC pluggable data source (PDS) enables you to access any JDBC sources, such as:

- An RDBMS like Oracle, DB2, Sybase, or SQL Server

- A non-relational data source like Microsoft Excel

- Any ODBC data source through the JDBC-ODBC bridge

The JDBC PDS is installed by default with Oracle Reports to allow access to all of the JDBC supported data sources.

This chapter contains the following sections:

- JDBC Configuration File

- Defining and Running a JDBC Query

- Troubleshooting Information

- Adding Your Own PDS

8.1 JDBC Configuration File

The `jdbcpds.conf` file, located in the `ORACLE_HOME`\reports\conf directory, is the Oracle Reports JDBC PDS configuration file. This file is pre-configured for the Merant DataDirect drivers provided by Oracle. You need to add or modify relevant entries in the `jdbcpds.conf` file to include any other JDBC drivers that you want to use.

Reports Builder displays a list of drivers in the JDBC Query Connection dialog box based on the entries in the `jdbcpds.conf` file. Use this list to select specific drivers for your report's JDBC query.

Reports Builder reads and caches the entries in the `jdbcpds.conf` when it is invoked. Restart Reports Builder to view the result of any changes made to the `jdbcpds.conf` file, e.g., adding a new JDBC driver entry.

The `jdbcpds.conf` file has two sections:

■ An Internal DTD section describing the XML format and driver configuration information

---

Caution: This section should not be modified.

■ An XML section detailing the driver information like driver name, connect string format, driver class, etc.,

---

Note: You can modify or add your driver information in this section.

Example

The following sample illustrates the contents of the `jdbcpds.conf` file:

```
<!-- DTD section - Not to be modified -->

<!DOCTYPE jdbcpds [
<!ELEMENT jdbcpds  (driverInfo)>
<!ELEMENT driverInfo (driver+)>
<!ELEMENT driver (property*)>
<!ATTLIST driver  name          CDATA  #REQUIRED
                  sourceDatabase (oracle   |
                                   sqlserver |
                                   sybase   |
                                    db2      |
                                    informix |
                                    odbc     |
                                    other) "oracle"
                  mainProtocol      ( jdbc ) "jdbc"
                  subProtocol   CDATA  #REQUIRED
                  connectString     CDATA  #REQUIRED
                  class         CDATA  #REQUIRED
                  connection    CDATA  #REQUIRED
                  loginTimeout      CDATA  "5"
>
<!ELEMENT property EMPTY>
```

```
<!ATTLIST property   name  CDATA  #REQUIRED
                     value CDATA  #REQUIRED >

]>


<!-- Add or modify the following section for your driver information -->
<!-- Following drivers are available out-of-box in 9iAS -->

<jdbcpds>
 <driverInfo>
  <driver name = "oracleThin"
          sourceDatabase = "oracle"
          subProtocol = "oracle:thin"
          connectString = "mainProtocol:subProtocol:@databaseName"
          class= "oracle.jdbc.driver.OracleDriver"
          connection = "oracle.reports.plugin.datasource.jdbcpds.
          JDBCConnectionHandling">
  </driver>

  <driver name = "oracle"
          sourceDatabase = "oracle"
          subProtocol = "oracle:oci8"
          connectString = "mainProtocol:subProtocol:@databaseName"
          class = "oracle.jdbc.driver.OracleDriver"
          connection = "oracle.reports.plugin.datasource.jdbcpds.
          JDBCConnectionHandling">
  </driver>

  <driver name = "jdbc-odbc"
          sourceDatabase = "odbc"
          subProtocol = "odbc"
          connectString = "mainProtocol:subProtocol:databaseName"
          class = "sun.jdbc.odbc.JdbcOdbcDriver"
          connection = "oracle.reports.plugin.datasource.jdbcpds.
          JDBCConnectionHandling">
  </driver>

  <driver name = "sqlserver-merant"
          sourceDatabase = "sqlserver"
          subProtocol = "merant:sqlserver"
          connectString = "mainProtocol:subProtocol://databaseName"
          class = "com.oracle.ias.jdbc.sqlserver.SQLServerDriver"
          connection = "oracle.reports.plugin.datasource.jdbcpds.
          JDBCConnectionHandling">
```

```
        </driver>

        <driver name = "sybase-merant"
                sourceDatabase = "sybase"
                subProtocol = "merant:sybase"
                connectString = "mainProtocol:subProtocol://databaseName"
                class = "com.oracle.ias.jdbc.sybase.SybaseDriver"
                connection = "oracle.reports.plugin.datasource.jdbcpds.
                JDBCConnectionHandling"
                loginTimeout = "0">
        </driver>

        <driver name = "db2-merant"
                sourceDatabase = "db2"
                subProtocol = "merant:db2"
                connectString = "mainProtocol:subProtocol://databaseName"
                class = "com.oracle.ias.jdbc.db2.DB2Driver"
                connection = "oracle.reports.plugin.datasource.jdbcpds.
                JDBCConnectionHandling"
                loginTimeout = "0">
        </driver>

        <driver name = "informix-merant"
                sourceDatabase = "informix"
                subProtocol = "merant:informix"
                connectString = "mainProtocol:subProtocol://databaseName"
                class = "com.oracle.ias.jdbc.informix.InformixDriver"
                connection = "oracle.reports.plugin.datasource.jdbcpds.
                 JDBCConnectionHandling">
        </driver>

  </driverInfo>
 </jdbcpds>
```

Table 8–1 outlines the various attributes that can be associated with a driver.

Table 8–1   Driver Attributes

Attribute Name	Description	Sample
name	A unique user-defined value used to refer to a specific JDBC driver in Oracle Reports.	sybase-merant

Table 8–1   Driver Attributes

Attribute Name	Description	Sample
sourceDatabase	Database referenced by the driver. The valid entries are: oracle sqlserver sybase db2 informix odbc other	oracle
subProtocol	Driver sub protocol added with the database URL before creating a database connection. This is driver-specific information and can be found in the driver documentation. Example: The sub protocol used for connecting to the Merant driver: Sybase is merant:sybase SQL Server is merant:sqlserver	merant:sybase
connectString	Format of the driver's connect string format is mainProtocol:subProtocol://databaseURL. For example, *jdbc:subProtocol://databaseName*. Do not specify the actual values for subProtocol or databaseName, use the fixed placeholder names instead.	mainProtocol:subProtocol://databaseName
class	Driver class name used to register to REPORTS_CLASSPATH and load the driver. This is driver-specific information and can be found in the driver documentation.	com.oracle.ias.jdbc.informix.InformixDriver

Table 8–1   Driver Attributes

Attribute Name	Description	Sample
connection	Driver's connection handling class. The JDBC PDS can have different connection handling classes for each driver. Oracle Reports' default connection handling class, which is sufficient for most drivers, is *oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling* Refer to the Oracle Reports Java document, for more information on how to extend your JDBC Connection class	`oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling`
loginTimeout (Optional)	Driver-specific parameter. Specify the value in seconds. Please refer to the driver documentation for more information.	0
property	Specify any additional properties of your driver as *Attribute Name* and *Value*.	–

When you submit your report's connection details, the connection information is combined with the driver's configuration information specified in the `jdbcpds.conf` file. The resulting connection information is submitted to the database as a complete connection URL. Refer to Table 8–2, Table 8–3, Table 8–4, Table 8–5, and Table 8–6 for more information on sample connection information.

Figure 8–1 shows a list of all drivers configured in the `jdbcpds.conf` file.

Figure 8–1   JDBC Connect dialog in Reports Builder
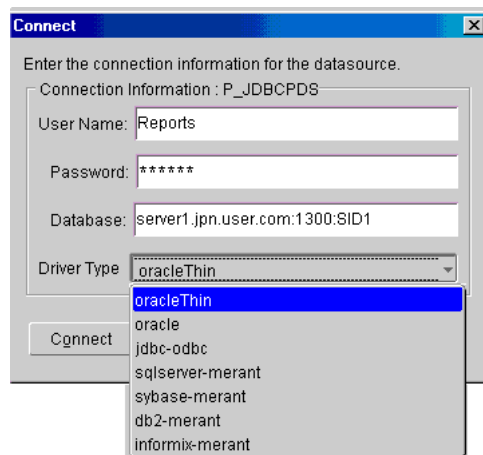
8.1.1 Verifying Pre-installed Driver Entries

Drivers like SQL Server and Excel with JDBC-ODBC, Oracle JDBC Thin, and Oracle JDBC OCI (thick) are installed and configured with Oracle Reports. These drivers do not require any additional JAR files to be installed.

- Oracle JDBC Thin driver

- Oracle JDBC OCI (thick) driver

- JDBC-ODBC driver

    You can use SQL Server / Excel with the JDBC-ODBC driver. This entry is pre-configured in the `jdbcpds.conf` file. Before you can use SQL Server or Excel with JDBC-ODBC, you need to create an ODBC data source. Refer to Windows help, for more information on how to create an ODBC data source.

    > **Note:**   Oracle Application Server provides Merant DataDirect drivers which can also be used to access SQL Server.

8.1.2 Installing and Configuring Merant DataDirect Drivers

Oracle provides a set of Merant DataDirect drivers (Version 3.2) that can be downloaded from OTN, (`http://otn.oracle.com`). The driver configuration

file, i.e., `jdbcpds.conf` contains relevant entries for the Merant DataDirect drivers. Additionally, the JDBC Connect dialog (Table 8–1) lists the entries for the set of Merant DataDirect drivers provided by Oracle.

However, you need to install the appropriate JAR files and specify them in Oracle Reports specific classpath entries, in order to make them available to Reports Builder and OracleAS Reports Services

The drivers provided by Oracle for use with Oracle Application Server / Oracle Developer Suite are:

- Sybase Driver
- DB2 Driver
- SQL Server Driver
- Informix Driver

You can also install and configure a Custom Driver for use with Oracle Application Server and Oracle Developer Suite.

The following procedure outlines the generic steps involved in configuring the Merant DataDirect drivers. To configure specific Merant DataDirect drivers refer to the appropriate sections.

To configure the Merant DataDirect drivers:

1. Install the relevant JAR files in your Oracle Application Server and Oracle Developer Suite directory.

2. Include an entry in the REPORTS_CLASSPATH to make the files available to Reports Builder and OracleAS Reports Services. Refer to the relevant driver in this section for information on the required JAR files.

   a. **Reports Builder**: Prefix the driver location to the existing entries in REPORTS_CLASSPATH. This variable is located in the registry for Windows users and in the `reports.sh` file for Solaris users. Refer to the relevant driver in this section for an example.

   b. **rwbuilder.conf**: Append the driver location to the engine `classPath` attribute in the `rwbuilder.conf` configuration file. Refer to the relevant driver in this section for an example.

   c. **Reports Server:** Append the driver location to the `classPath` attribute of the engine, in the Reports Server configuration file. Refer to the relevant driver in this section for an example

d. **jdbcpds.conf**: Located in the *ORACLE_HOME*\reports\conf directory. Refer to Table 8–1 for more information on the parameters. Refer to the relevant driver in this section for an example.

8.1.2.1 Sybase Driver

1. Install the relevant JAR files in your Oracle Application Server and Oracle Developer Suite directory.

    Jar files required: YMutil.jar, YMsybase.jar, and YMbase.jar.

2. Include an entry in the REPORTS_CLASSPATH to make the files available to Reports Builder and OracleAS Reports Services.

    a. **Reports Builder**: Prefix the driver location to the existing entries in REPORTS_CLASSPATH. This variable is located in the registry for Windows users and in the reports.sh file for Solaris users.

    Example:
    ```
    D:\sybase_installed\YMutil.jar;D:\sybase_
    installed\YMsybase.jar;D:\sybase_installed\YMbase.jar;existing classpath
    entries
    ```

    b. **rwbuilder.conf**: Append the driver location to the engine classPath attribute in the rwbuilder.conf configuration file.

    Example:
    ```
    <engine id="rwEng" class="oracle.reports.engine.EngineImpl"
    initEngine="1"  maxEngine="1"  minEngine="0" engLife="50" maxIdle="30"
    callbackTimeOut="60000"     classPath="D:\sybase_
    installed\YMutil.jar;D:\sybase_installed\YMsybase.jar;D:\sybase_
    installed\YMbase.jar;">
    ...
    </engine>
    ```

    c. **Reports Server:** Append the driver location to the classPath attribute of the engine in the Reports Server configuration file.

    Example:
    ```
    <engine id="rwEng" class="oracle.reports.engine.EngineImpl"
    initEngine="1"  maxEngine="1"  minEngine="0" engLife="50" maxIdle="30"
    callbackTimeOut="60000"     classPath="D:\sybase_
    installed\YMutil.jar;D:\sybase_installed\YMsybase.jar;D:\sybase_
    installed\YMbase.jar;">
    ...
    </engine>
    ```

d. **`jdbcpds.conf`**: Located in the *ORACLE_HOME*\reports\conf directory. Refer to Table 8–1 for more information on the required parameters.

Example:
```
<driver   name = "sybase-merant"
          sourceDatabase = "sybase"
              subProtocol = "merant:sybase"
              connectString = "mainProtocol:subProtocol://databaseName"
              class = "com.oracle.ias.jdbc.sybase.SybaseDriver"
              connection = "oracle.reports.plugin.datasource.jdbcpds.
              JDBCConnectionHandling"
              loginTimeout = "0">
   </driver>
```

8.1.2.2 DB2 Driver

1. Install the relevant JAR files in your Oracle Application Server and Oracle Developer Suite directory.

JAR files required: YMutil.jar, YMdb2.jar,and YMbase.jar

2. Include an entry in the REPORTS_CLASSPATH to make the files available to Reports Builder and OracleAS Reports Services.

a. **Reports Builder**: Prefix the driver location to the existing entries in REPORTS_CLASSPATH. This variable is located in the registry for Windows users and in the reports.sh file for Solaris users.

Example:
```
D:\db2_installed\YMutil.jar;D:\db2_installed\YMdb2.jar;D:\db2_
installed\YMbase.jar;existing classpath entries
```

b. **rwbuilder.conf**: Append the driver location to the engine classPath attribute in the rwbuilder.conf configuration file.

Example:
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\db2_
installed\YMutil.jar;D:\db2_installed\YMdb2.jar;D:\db2_
installed\YMbase.jar">
...
</engine>
```

c. **Reports Server:** Append the driver location to the classPath attribute of the engine in the Reports Server configuration file.

Example:
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\db2_
installed\YMutil.jar;D:\db2_installed\YMdb2.jar;D:\db2_
installed\YMbase.jar">
...
</engine>
```

    d. **jdbcpds.conf**: Located in the *ORACLE_HOME*\reports\conf directory. Refer to Table 8–1 for more information on the parameters.

Example:
```
<driver   name = "db2-merant"
                sourceDatabase = "db2"
                subProtocol = "merant:db2"
                connectString = "mainProtocol:subProtocol://databaseName"
                class = "com.oracle.ias.jdbc.db2.DB2Driver"
                connection = "oracle.reports.plugin.datasource.jdbcpds.
                JDBCConnectionHandling"
                loginTimeout = "0">
    </driver>
```

8.1.2.3 SQL Server Driver

1. Install the relevant .jar files in your Oracle Application Server and Oracle Developer Suite directory.

Jar files required: YMutil.jar, YMsqlserver.jar, and YMbase.jar

2. Include an entry in the REPORTS_CLASSPATH to make the files available to Reports Builder and OracleAS Reports Services.

    a. **Reports Builder**: Prefix the driver location to the existing entries in REPORTS_CLASSPATH. This variable is located in the registry for Windows users and in the reports.sh file for Solaris users.

Example:
```
D:\sqlserver_installed\YMutil.jar;D:\sqlserver_
installed\YMsqlserver.jar;D:\sqlserver_installed\YMbase.jar;existing
classpath entries
```

    b. **rwbuilder.conf**: Append the driver location to the engine classPath attribute in the rwbuilder.conf configuration file.

Example:

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\sqlserver_
installed\YMutil.jar;D:\sqlserver_
installed\YMsqlserver.jar;D:\sqlserver_installed\YMbase.jar;">
...
</engine>
```

c. **Reports Server:** Append the driver location to the classPath attribute of the engine in the Reports Server configuration file.

Example:
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\sqlserver_
installed\YMutil.jar;D:\sqlserver_
installed\YMsqlserver.jar;D:\sqlserver_installed\YMbase.jar;">
...
</engine>
```

d. **jdbcpds.conf**: Located in the *ORACLE_HOME*\reports\conf directory. Refer to Table 8–1 for more information on the parameters.

Example:
```
<driver  name = "sqlserver-merant"
             sourceDatabase = "sqlserver"
             subProtocol = "merant:sqlserver"
             connectString = "mainProtocol:subProtocol://databaseName"
             class = "com.oracle.ias.jdbc.sqlserver.SQLServerDriver"
             connection = "oracle.reports.plugin.datasource.jdbcpds.
             JDBCConnectionHandling">
</driver>
```

8.1.2.4 Informix Driver

1. Install the relevant JAR files in your Oracle Application Server and Oracle Developer Suite directory.

JAR files required: YMutil.jar, YMinformix.jar, and YMbase.jar

2. Include an entry in the REPORTS_CLASSPATH to make the files available to Reports Builder and OracleAS Reports Services.

a. **Reports Builder**: Prefix the driver location to the existing entries in REPORTS_CLASSPATH. This variable is located in the registry for Windows users and in the reports.sh file for Solaris users.

Example:
```
D:\informix_installed\YMutil.jar;D:\informix_
installed\YMinformix.jar;D:\informix_installed\YMbase.jar;existing
classpath entries
```

b. **rwbuilder.conf**:Append the driver location to the engine `classPath`
    attribute in the `rwbuilder.conf` configuration file.

Example:
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\informix_
installed\YMutil.jar;D:\informix_installed\YMinformix.jar;D:\informix_
installed\YMbase.jar">
...
</engine>
```

c. **Reports Server:** Append the driver location to the `classPath` attribute of
    the engine in the Reports Server configuration file.

Example:
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\informix_
installed\YMutil.jar;D:\informix_installed\YMinformix.jar;D:\informix_
installed\YMbase.jar">
...
</engine>
```

d. **jdbcpds.conf**: Located in the *ORACLE_HOME*\reports\conf directory.
    Refer to Table 8–1 for more information on the parameters.

Example:
```
<driver name = "informix-merant"
      sourceDatabase = "informix"
      subProtocol = "merant:informix"
      connectString = "mainProtocol:subProtocol://databaseName"
      class = "com.oracle.ias.jdbc.informix.InformixDriver"
      connection = "oracle.reports.plugin.datasource.jdbcpds.
      JDBCConnectionHandling">
</driver>
```

8.1.2.5 Custom Driver
Any driver that is not provided by Oracle must be installed and configured:

1. Install the relevant JAR files in your Oracle Application Server and Oracle Developer Suite directory.

2. Include an entry in the REPORTS_CLASSPATH to make the files available to Reports Builder and OracleAS Reports Services.

   Jar files required: Refer to the relevant driver documentation.

   a. **Reports Builder**: Prefix the driver location to the existing entries in REPORTS_CLASSPATH. This variable is located in the registry for Windows users and in the reports.sh file for Solaris users.

      Example:
      ```
      driver location\1st jar file;driver location\2nd jar file2;existing
      classpath entries
      ```

   b. rwbuilder.conf: Append the driver location to the engine classPath attribute in the rwbuilder.conf configuration file.

      Example:
      ```
      <engine id="rwEng" class="oracle.reports.engine.EngineImpl"
      initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
      maxIdle="30" callbackTimeOut="60000" classPath="driver location\1st jar
      file;driver location\2nd jar file;">
      ...
      </engine>
      ```

   c. **Reports Server**: Append the driver location to the classPath attribute of the engine in the Reports Server configuration file.

      Example:
      ```
      <engine id="rwEng" class="oracle.reports.engine.EngineImpl"
      initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
      maxIdle="30" callbackTimeOut="60000" classPath="driver location\1st jar
      file;driver location\2nd jar file;">
      ...
      </engine>
      ```

   d. **jdbcpds.conf**: Located in the *ORACLE_HOME*\reports\conf directory. Add relevant driver configuration information to the jdbcpds.conf file. Refer to Table 8–1 for more information on the required parameters.

      Example:
      ```
      <driver  name = "<driver name>"
            sourceDatabase = "<sourceDatabase>"
                  subProtocol = "<subProtocol>"
                  connectString = "mainProtocol:subProtocol://databaseName"
      ```

```
                 class = "<driver class name>"
                 connection ="<connection handling class">
</driver>
```

Note: This value can still be `connection = "oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling"` for your custom drivers, if you do not want to implement a custom connection dialog

---

8.2 Defining and Running a JDBC Query

After configuring the relevant JDBC drivers, you can define and run a JDBC query using either SQL or a stored procedure.

To define a JDBC query:

1. Start Reports Builder.

2. Invoke the Reports Wizard.

3. Select the data source type as **JDBC Query** and click **Next**. For more information on how to work with the Report Wizard, refer to the *Reports Builder online help*.

Figure 8–2   Select a Data Source Type
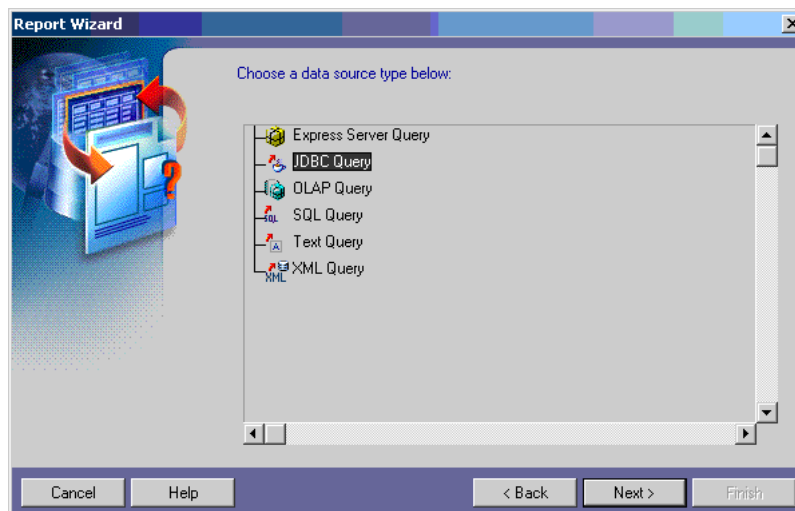
4. In the Data Source Definition window, click **Query Definition**.

5. Define one of the following:

   - A SQL query:

     ```
     SELECT * FROM DEPARTMENT;
     ```

   - A stored procedure:

     Enter the complete call syntax of your database's stored procedure. For example:

     ```
     TestProc(40)
     ```

     For more information on the call syntax, refer to your database documentation.

     JDBC PDS submits the calling statement to the driver as specified, to invoke the stored procedure.
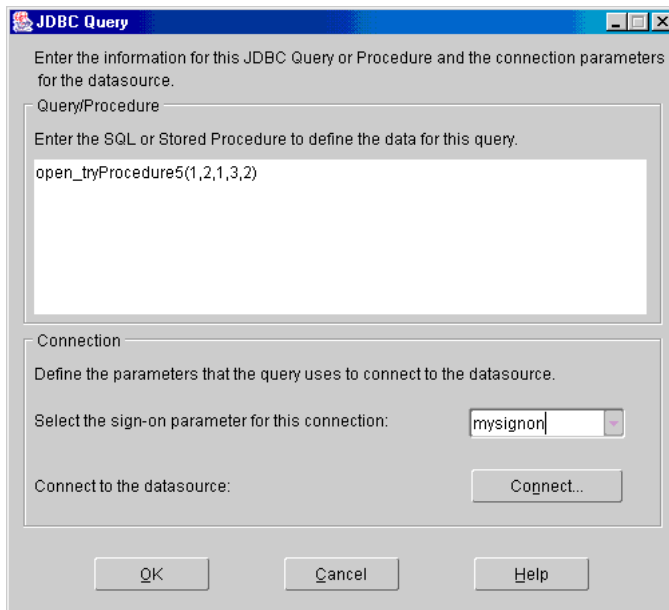
Usage Notes

- To specify an Excel data source:

- Query (Single worksheet):

  SELECT * FROM [SHEET1$] **or** SELECT COL1, COL2, ...COLn
  FROM [SHEET1$]

  * Where SHEET1$ is the name of a .xls file

  * Where the first worksheet row value is taken as a column name for the
    query

- Query (Multiple worksheets):

  SELECT * FROM [WORKSHEETNAME$]

  * Where [WORKSHEETNAME$] is the name of the worksheet

  * Where the first worksheet row is taken as a column name for the query

---

Note: If a value is not mentioned in any of the columns in the first
row, then the default name is *FcolumnNumber*. For example, the
8th column will be F8, the ninth column will be F9, and so on.

---

6. Specify a sign-on parameter name. This sign-on parameter is associated with
   the connection information when run against a database. The default sign-on
   parameter value is p_jdbcpds:

   a. Enter a new sign-on name and click **Connect**. Use this sign-on parameter to
      specify a database connection when you are running your report using
      OracleAS Reports Services.

   b. Enter the connection information (user name, password, and database
      name) for the driver type. Refer to Table 8–2, Table 8–3, Table 8–4, Table 8–5,
      and Table 8–6 for sample connection information.

   c. Select the driver type. The driver list is displayed based on the values
      entered in the jdbcpds.conf file.

   d. Click **Connect** to gain access to the database using the new sign-on. The
      connect string formed internally is a combination of:

      * The connectString driver attribute (Table 8–1) defined in the
        jdbcpds.conf file

      * The connection information supplied in the Connect dialog.

7. Click **OK** to execute the JDBC query.

8. The Reports Wizard displays the query description.
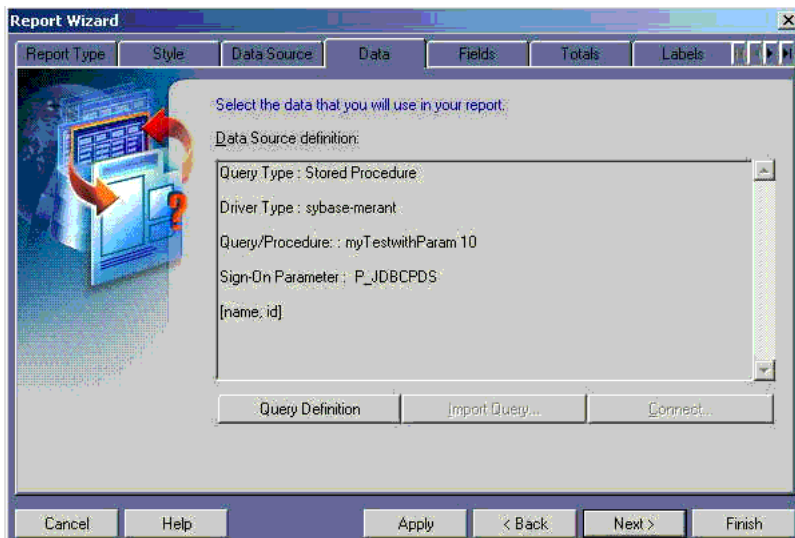
9. Follow the steps in the wizard to define the layout and to run the report based on your JDBC query.

8.2.1 Sample Connection Information

Table 8–2, Table 8–3, Table 8–4, Table 8–5, and Table 8–6 lists sample connection information for use with the pre-installed drivers.

Table 8–2   Oracle Thin Driver

Property	Value
Username	Reports
Password	Welcome

Table 8–2   Oracle Thin Driver

Property	Value
Database	`hostname:` The TCP/IP address or TCP/IP host name of the server you are connecting to.
	`port:` The TCP/IP port number.
	`property:` The connection properties. Refer to the driver documentation for a list of connection properties and their valid values.
	Example: `server1.us.oracle.com:1300:session1`

Table 8–3   Oracle Thick Driver

Property	Value
Username	`Reports`
Password	`Welcome`
Database	`n123`
	where `n123` is a tnsname entry in the `tnsnames.ora` file

Table 8–4   JDBC-ODBC Driver

Property	Value
Username	`N/A`
Password	This password is set at the time of establishing an ODBC connection.
Database	`SQLSVR`
	where `SQLSVR` is the ODBC Data entry in the ODBC data source

Table 8–5   Sybase

Property	Value
Username	`Reports`
Password	`Welcome`

Table 8–5 Sybase

Property	Value
Database	`hostname`: The TCP/IP address or TCP/IP host name of the server you are connecting to.
	`port`: The number of the TCP/IP port.
	Example: `server1.us.oracle.com:1300`

Table 8–6 DB2

Property	Value
Username	`Reports`
Password	`Welcome`
Database	`hostname`: The TCP/IP address or TCP/IP host name of the server you are connecting to.
	`port`: The TCP/IP port number.
	`property`: The connection properties. Refer to the driver documentation for a list of connection properties and their valid values.
	Example1: `server1:1654`
	Example2: `server2:1721;PackageName=pkg1`

8.3 Running a JDBC Report Using OracleAS Reports Services

When you run a report having a JDBC query (Reports Server or `rwrun` engine), use the sign-on parameter to submit the connection information for the JDBC data source. This sign-on parameter is defined for your JDBC query in the design time.

For example, if your report has a JDBC query to a Sybase data source, a JDBC query to a DB2 data source, and a SQL query to an Oracle data source, then the request could be defined as:

```
http://your ias
server:port//reports/rwservlet?report=my.rdf&userid=user/pwd@oracledb
&desformat-pdf&destype=cache&p_sybasepds=sybaseuser/pw@sybasehost:port
&p_db2pds=db2user/pwd@db2host:port
```

where:

- `p_sybasepds` is the sign-on parameter associated with the sybase JDBC query.

- ■ `p_db2pds` is the sign-on parameter associated with the DB2 JDBC query defined in the report at design time.

- ■ `userid` is the value for connecting the SQL query to the Oracle database. You do not need to specify the `userid` if your report does not have a SQL query or a REF cursor query.

The default sign-on parameter name `p_jdbcpds` will be used if you have not specified a name in the JDBC query dialog while designing the report.

8.4 Troubleshooting Information

This section lists:

- ■ JDBC PDS error messages (Error Messages)

- ■ JDBC query troubleshooting (Trace Information).

8.4.1 Error Messages

Table 8–7, Table 8–8, and Table 8–9 lists troubleshooting information related to the JDBC PDS.

Table 8–7   Error Messages related to the database connection

Error Message	Cause	Action
Connection class {0} can't be loaded	Invalid connection class mentioned in the `jdbcpds.conf` file for the selected driver.	Ensure that the driver connection class specified in the `jdbcpds.conf` file is both valid and available.
Failed to connect to the datasource	Invalid connection information.	Ensure the validity of the username, password, database, and driver type.
Invalid sign-on parameter {0}	Invalid sign-on parameter for the specified query or procedure.	Ensure the sign-on parameter is available and valid for the report's JDBC query type.
Invalid value is given to the sign-on parameter {0}.	Invalid connect string for the specified sign-on parameter.	Ensure that the specified connect string for this sign-on parameter is valid for the selected driver.

Table 8–8   Error messages related to executing the data source

Error Message	Cause	Action
Reference parameter of type Date is not supported by JDBC driver used.	The driver used to connect to database does not support the **Date** data type as a reference parameter.	Use either: The **String** data type as the reference parameter A different JDBC driver that supports the **Date** data type as a reference parameter.
Invalid lexical parameter {0} is used in the query	Invalid lexical parameter used in the query or procedure.	Ensure that the query or procedure uses valid lexical parameters. Create a new parameter if it is not available.
SQL Error:	SQL syntax error in the specified query or procedure.	Ensure that the syntax of the query or procedure is valid. Refer to the relevant data source's documentation.
Invalid query/procedure for the specified datasource.	Invalid query or procedure syntax.	Ensure that the syntax of the query or procedure is valid. Refer to the relevant data source's documentation
Invalid reference parameter value	Invalid reference parameter value.	Verify that the reference column types and values are correct.
No query/procedure is entered.	The query or procedure text field is empty.	Enter a valid query or procedure in the text field.
Database URL:	Invalid database URL.	Verify the validity of the specified database name and the selected driver type.
Either the number of columns or the types of columns does not match the query definition	The data fetched does not match the number of columns or column types specified in the query definition.	Ensure that the number of columns and the column types match the query definition.

Table 8–8 Error messages related to executing the data source

Error Message	Cause	Action
`The column type {0} used in the query/procedure is not supported by Reports JDBC query.`	This column type is not supported by the Oracle Reports JDBC query interface.	Ensure that only column types supported by the Oracle Reports JDBC query interface are used. Refer to the JDBC specification and Oracle Reports documentation for a list of all supported types.

Table 8–9 Isolating driver / pds issues

Error Message	Cause	Action
`The inline DTD section of the configuration file jdbcpds.conf has been modified.`	The format of the inline DTD section in the `jdbcpds.conf` file has been altered.	If the DTD format is modified, ensure the validity of configuration file against the JDBC PDS requirement.
`Line Number:`	An error was found on the specified line of the `jdbcpds.conf` file.	Correct the error on the specified line.
`Configuration file jdbcpds.conf is not found`	The `jdbcpds.conf` file is not found under the `reports/conf` directory.	Ensure that the `jdbcpds.conf` file is available in the `reports/conf` directory.
`Parsing error in the configuration file jdbcpds.conf. Number of errors:{0}`	The XML section in the `jdbcpds.conf` file does not conform with its inline DTD.	Ensure that the XML section in the `jdbcpds.conf` file refers to the correct inline DTD.
`No entry is present for the driver {0} in the jdbcpds.conf file.`	The driver used in the query is not mentioned in the `jdbcpds.conf` file.	Ensure that the entry for the required driver along with the related driver information is in the `jdbcpds.conf` file.

8.4.2 Trace Information

Use the detailed trace information (`ORACLE_HOME\reports\logs\`) generated by Oracle Reports to debug your JDBC query.

- Design time (building a JDBC query) and run time (running a JDBC query)

The trace information generated is helpful to find out the following:

- Lexical and bind parameters.

- Final connect string formed to connect to the driver.

- Metadata information received from the driver.

- Final query submitted to the database.

See Example 8–1 for a sample design-time trace output.

See Example 8–2 for a sample run-time trace output.

Sample trace output

Example 8–1   Building a JDBC Query from JDBC Query Dialog

Connection handling trace showing final connect string
```
[2003/4/7 5:41:38:686] Debug 50103 (jdbcpds): handleConnectButtonEvent : start
[2003/4/7 5:41:38:686] Debug 50103 (jdbcpds): handleConnectButtonEvent :
subProtocol :sybase-merant
[2003/4/7 5:41:38:686] Debug 50103 (jdbcpds): handleConnectButtonEvent :
connection class
:oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling
[2003/4/7 5:41:38:696] Debug 50103 (jdbcpds): handleConnectButtonEvent : combine
string :jdbc:merant:sybase://server1.us.oracle.com:1300
[2003/4/7 5:41:38:696] Debug 50103 (jdbcpds): JDBCDataSource : setJDBCQueryType:
sybase
[2003/4/7 5:41:41:350] Debug 50103 (jdbcpds): JDBCUIEventHandler :
handleConnectEvent : Valid Connection
com.oracle.ias.jdbc.sybase.SybaseConnection@56fc16
[2003/4/7 5:41:41:350] Debug 50103 (jdbcpds): JDBCUIEventHandler :
handleConnectEvent : END com.oracle.ias.jdbc.sybase.SybaseConnection@56fc16
```

Design time metadata of query
```
[2003/3/31 6:35:46:363] Debug 50103 (jdbcpds): JDBCUIEventHandler :
handleOKEvent : Serialize XML<jdbcpds DTDVersion="
1.0"><JDBCQuery>jdbcpdspkg.proc_with_
param(1,2,3,4,5)</JDBCQuery><QueryDefinition>1</QueryDefinition><driverType>orac
le</driverType><connectionClass>oracle.reports.plugin.datasource.jdbcpds.JDBCCon
nectionHandling</connectionClass><SignOnParameter>P_
JDBCPDS</SignOnParameter><jdbcElements><elementname = "EMPNO"  type = "2"
typeName = "NUMBER"  columnSize = "4"  columnScale = "0" /><element name =
"ENAME"  type = "12"   typeName ="VARCHAR2"  columnSize = "10"  columnScale =
"0" /><element name = "JOB"  type = "12"   typeName = "VARCHAR2"  columnSize =
"9"  columnScale ="0" /><element name = MGR"  type = "2"   typeName = "NUMBER"
```

```
columnSize = "4"  columnScale = "0" /><element name = "HIREDATE"  type = "93"
typeName = "DATE"  columnSize = "16"  columnScale = "0" /><element name = "SAL"
type = "2"   typeName = "NUMBER"  columnSize = "7"  columnScale= "2" /><element
name = "COMM"  type = "2"   typeName = "NUMBER"  columnSize = "7"  columnScale =
"2" /><element name = "DEPTNO"  type = "2" typeName = "NUMBER"  columnSize = "2"
columnScale = "0"
/></jdbcElements><referenceColumns></referenceColumns></jdbcpds>
[2003/3/31 6:35:46:383] Debug 50103 (jdbcpds): JDBCUIEventHandler :handleOKEvent
END
```

Example 8–2   Running a JDBC Query:

```
[2003/3/18 5:45:17:707] Debug 50103 (jdbcpds): JDBCDataSource : startRuntime
method : START
```

Describing the JDBC Query:
```
[2003/3/18 5:45:17:707] Debug 50103 (jdbcpds):  JDBCDataSource : describe :
START
[2003/3/18 5:45:17:707] Debug 50103 (jdbcpds):  applyXML: Extract the Serilzed
XML containing Query Meta Data <jdbcpds DTDVersion=" 1.0"><JDBCQuery>select *
from
emp</JDBCQuery><QueryDefinition>0</QueryDefinition><driverType>oracle</driverTyp
e><connectionClass>oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandli
ng</connectionClass>...
```

ConnectionHandling At Runtime:
```
[2003/3/18 5:45:17:737] Debug 50103 (jdbcpds): JDBCDataSource : startRuntime :
Create a new connection and handle it
[2003/3/18 5:45:17:737] Debug 50103 (jdbcpds):  JDBCExecuteQuerySource :
handleConnection : START
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
handleConnection : set driver
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
handleConnection : Check if Connection for the sign on parameter is pooled
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCExecuteQuerySource
:handleConnection :   connection available in pool
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): handleConnection : END
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCDataSource : startRuntime :
END
```

Runtime execution of jdbc query
```
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCDataSource : execute : run
Query
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
```

```
getOutputFromDatabase : START
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase: start Query stringto be submitted
jdbcpdspkg.proc_with_param(1,2,3,4,5)
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : check connection
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : QSource Id: 1
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure:Start
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure:Procedure to be submitted { call
jdbcpdspkg.proc_with_param(?,?,?,?,?,?) }
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure:  Set parameters for the procedure call
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure:  execute procedure
[2003/3/31 6:36:2:847] Debug 50103 (jdbcpds): JDBCDataSource : execute : query
execution over andresulset object is
oracle.jdbc.driver.OracleResultSetImpl@751a9e
[2003/3/31 6:36:2:847] Debug 50103 (jdbcpds): JDBCDataSource : execute : END
```

Running Report trace with Result set info

```
2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCDataSource : execute : replace
lexical columns withactual string for the query
[2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCDataSource : execute : run
Query
[2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : START
[2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase: start Query stringto be submitted select * from reports
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : check connection
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : QSource Id: 4
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : Query source is SQL query
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:executeQuery
Start
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): executeQuery prepareStatement select
* from reports
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): executeQuery : bind parameters set
for the query
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): executeQuery : JDBC Query executed
```

```
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : Query result  col 0  test col 1  10
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:executeQuery
Start
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): executeQuery prepareStatement
select * from reports
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): executeQuery : bind parameters set
for the query
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): executeQuery : JDBC Query executed
[2003/4/7 5:26:7:767] Debug 50103 (jdbcpds): JDBCDataSource : execute : query
execution over andresulset object is
com.oracle.ias.jdbc.base.BaseResultSet@56c3cf
[2003/4/7 5:26:7:767] Debug 50103 (jdbcpds): JDBCDataSource : execute : END
```

8.5 Adding Your Own PDS

Oracle Reports has exposed the PDS API and also contains an RSDK tutorial
(http://otn.oracle.com/products/reports/apis/index.html) that
describes in detail how to implement or customize your own PDS. Using this API,
you can implement an unlimited number of PDSs to access any kind of data sources
that you have.

The main tasks you must perform to add your PDS are:

- Registering the PDS
- Configuring the jdbcpds.conf File
- Installing Your PDS JAR Files
- Installing the Driver's JAR Files

8.5.1 Registering the PDS

Register your PDS with the preferences file to make it available to Reports Builder
and OracleAS Reports Services.

Table 8–10 Preferences file

File	Location	Operating System
cauprefs.ora	*ORACLE_HOME*	Windows
prefs.ora	*ORACLE_HOME*	Unix

The preferences file should reference the factory class implementation of either Oracle Reports or your custom class file. Reports Builder displays the relevant PDS icon only if the factory class is registered:

```
Reports.PluggableDataSourceFactories =
 ("oracle.reports.plugin.datasource.xmlpds.XMLDataSourceFactory",
  "oracle.reports.plugin.datasource.jdbcpds.JDBCDataSourceFactory",
  "oracle.reports.plugin.datasource.textpds.TextDataSourceFactory",
  "oracle.reports.plugin.datasource.myPDS.myPDSDataSourceFactory",
  "oracle.dss.pds.bibeans.xrpds.XRPDSFactory",
  "oracle.dss.pds.snapi.expresspds.ExpressPDSFactory")
```

To add your PDS:

1. Open the preferences file (Table 8–10) using any text editor.

2. Locate `Reports.PluggableDataSourceFactories`.

3. Include the name of any JDBC factory class using the following syntax:

   ```
   Reports.PluggableDataSourceFactories = ("pluginClassname"[,
   "pluginClassname"]...)
   ```
   `pluginClassname` is the name of the factory class containing the required implementation.

   Note: Refer to the Oracle Reports SDK on OTN, (`http://otn.oracle.com`), for more information on implementing your custom PDS class.

   ---

4. Restart Reports Builder / OracleAS Reports Services for the changes to take effect.

8.5.2 Configuring the jdbcpds.conf File

For more information on how to configure the `jdbcpds.conf` file, refer to the JDBC Configuration File section.

8.5.3 Installing Your PDS JAR Files

Specify the path to your PDS JAR files. This makes all the relevant classes available to Reports Builder and Reports Server.

8.5.3.1 Reports Builder

Prefix the path of all dependent JAR files to the Oracle Reports environment variable REPORTS_CLASSPATH. This variable is located in the registry for Windows users and in the reports.sh file for Solaris users.

Example: `D:\mypds.jar;`*existing classpath entries*

8.5.3.2 rwbuilder.conf

Add the JAR file location to the engine classPath attribute in the rwbuilder.conf configuration file.

Example:
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
    initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
    maxIdle="30" callbackTimeOut="60000" classPath="d:\mypds.jar;">
    ...
    </engine>
```

8.5.3.3 Reports Server

Append the jar file location to the classPath attribute in the Reports Server configuration file.

Example:
```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
    initEngine="1"  maxEngine="1" minEngine="0" engLife="50"
    maxIdle="30" callbackTimeOut="60000" classPath="d:\mypds.jar;">
    ...
```
</engine>

8.5.4 Installing the Driver's JAR Files

Refer to the Custom Driver section for more information on how to install the driver's JAR files.

9

Securing OracleAS Reports Services

The celebrated openness of the Internet brings with it concerns about controlling who has access to what confidential company information. OracleAS Reports Services provides a number of security options that enable you to ensure that the appropriate users are getting important data in a secure fashion. This chapter provides an overview of the available security options.

- About OracleAS Reports Services Security
- Configuring OracleAS Reports Services Security

9.1 About OracleAS Reports Services Security

This section describes how OracleAS Reports Services security operates to secure access to your reports and the data they include.

- Resources Protected
- Authorization and Access Enforcement
- Leveraging Oracle Identity Management Infrastructure

9.1.1 Resources Protected

OracleAS Reports Services encompasses functionality for three main areas of security:

- Application security (i.e., controlling access to the report application, where users launch report requests)
- Resource security (i.e., controlling access to reports, printers, calendars, and Reports Servers)
- Data source security (i.e., for controlling access to a particular database)

9.1.1.1 Application Security

Typically, users must log on to an application or site from which they can access and run their reports. This launcher application is typically protected by some sort of login facility, such as OracleAS Single Sign-On. Once they successfully gain entry into the launcher application, resource security takes over and determines which reports and destinations a given user or group may request.

For application security, OracleAS Single Sign-On provides a single point of user login and, optionally, data source security. In a typical configuration, the user would log on through OracleAS Single Sign-On to gain access to a report application, where they would access and run their reports.

Oracle Internet Directory stores user and group privilege information which is used by OracleAS Single Sign-On. Oracle Internet Directory also stores data source security information on a per user basis. Oracle Delegated Administration Services edits the information stored in Oracle Internet Directory. Oracle Delegated Administration Services can be accessed from within OracleAS Portal or separately, as a standalone component.

Alternatively, you might have your own application for launching reports with its own login mechanism and user/group repository. In this case, OracleAS Reports Services provides interfaces that allow you to integrate it with these non-Oracle components.

> **See Also:** Configuring OracleAS Reports Services Security
>
> For more information on these interfaces.

9.1.1.2 Resource Security

Resource security ensures that only authorized users or groups execute a specific report. It also keeps users or groups from accessing particular printers or Reports Servers for the execution of the report. You might well imagine a situation where certain printers and servers might be reserved for a particular group of users. Alternatively, some printers and servers may simply be inaccessible during certain times for maintenance activities.

Once it is determined that a user has the necessary privileges to execute a given report via the specified Reports Server to the specified destination, then the user's privileges to the data source accessed by the report must be ascertained.

OracleAS Portal provides resource security for reports, printers, calendars, and Reports Servers out of the box. In a typical configuration, the administrator or developer could specify which users and groups could access which reports, Reports Servers, and printers from OracleAS Portal.

As with application security, you might have your own mechanism for protecting resources. In this case, OracleAS Reports Services provides interfaces that allow you to integrate it with these non-Oracle components.

> **See Also:** Configuring OracleAS Reports Services Security
>
> For more information on these interfaces.

9.1.1.3 Data Source Security

Data source security defines the users or roles that can access the data within the given data source. A report might access multiple data sources and the current user must have privileges on all of the data sources accessed by the report in order to run it and view the output. The data source administrator (typically a DBA) grants access to data sources. Data source security must be established and in place prior to configuring your reports environment.

You can provide for data source security in two different ways with OracleAS Reports Services:

- You can associate data source connection information with a Single Sign-On user. In this case, the first time a user attempts to access the data source, Oracle Delegated Administration Services prompts them to create a resource for their data source connection. After the user creates this data source resource, OracleAS Single Sign-On associates it with the user in Oracle Internet Directory. Once the data source resource is associated with the Single Sign-On user, it becomes part of their Single Sign-On identity and they can access the data source without having to log into it separately. This method has two key advantages. First, it enables each user to gain access to the data source through their Single Sign-On identity without having to login separately. Second, it enables a single report URL to be used by many users because the data source login information is stored with the user's identity and therefore does not have to be hard coded into the report's URL or a key mapping.

- In your report URLs or key mappings, you can code AUTHID and the necessary connection parameters (e.g., USERID) for your report. This functionality is much the same as it was in previous releases of OracleAS Reports Services. For a complete discussion of URL syntax, refer to The Reports URL Syntax. For a complete discussion of key mapping, refer to Using a Key Map File.

As with the other security areas, you might have your own mechanism for protecting data sources. In this case, OracleAS Reports Services provides interfaces that allow you to integrate it with these non-Oracle components.

> **See Also:** Configuring OracleAS Reports Services Security
>
> For more information on these interfaces.

9.1.2 Authorization and Access Enforcement

Access control for report requests can be maintained with or without OracleAS Single Sign-On.

- Handling Report Requests with OracleAS Single Sign-On
- Handling Report Requests without OracleAS Single Sign-On

9.1.2.1 Handling Report Requests with OracleAS Single Sign-On

OracleAS Single Sign-On makes use of an encrypted cookie to track authenticated application users. When `rwservlet` receives a request to execute a report on a secured Reports Server, it queries the Oracle HTTP Server (via the getRemoteUser call) to determine whether the user has already logged on through OracleAS Single Sign-On (i.e., a Single Sign-On cookie exists for the user):

- If the user has logged on already (i.e., the cookie exists), then rwservlet gets the user's identity from the Oracle HTTP Server.
- If the user has not logged on already (i.e., the cookie does not exist yet), then the Oracle HTTP Server redirects the user to OracleAS Single Sign-On, which prompts the user to login. Once the user is authenticated, the Single Sign-On cookie is created and the user is redirected back to `rwservlet`, which then proceeds as described in the previous bullet item.

> **Note:** If the report request is launched from within OracleAS Portal rather than `rwservlet`, OracleAS Reports Services will similarly validate the user's privileges on the report before running it. Even for unauthenticated (PUBLIC) users viewing public pages, OracleAS Reports Services verifies that the PUBLIC user account has appropriate privileges on the report.

9.1.2.1.1 Report Request Flow with Single Sign-On In this scenario, a report request is sent to a secured Reports Server with Single Sign-On enabled.

1. If the report is to be run from within OracleAS Portal, the user must be logged into OracleAS Portal and, consequently, OracleAS Single Sign-On. As part of its security, OracleAS Portal validates that the user has the required security

permissions to see the report object. For example, if the report object is on a page, the user must have appropriate privileges to see the page and the reports object. Otherwise, OracleAS Portal will not display the page or the report object to the user.

If the report is not to be run from within OracleAS Portal, the user must somehow have gained access to the URL that launches the report request (e.g., via a link on a Web page or a bookmark).

2. The report request is made via one of the following methods:

- From within OracleAS Portal, the user requests to run the report object (e.g., clicks the Run link).

- From outside of OracleAS Portal, the user chooses a link or a bookmark that contains a URL that requests the report.

> **Note:** The URL may optionally contain or reference (i.e., via the key map file) a Single Sign-On parameter (`SSOCONN`) with a value of the form:
>
> `key_name/data_source_type/parameter_name`
>
> In the case of an Oracle database, the Single Sign-On value would look something like the following:
>
> `mykey/OracleDB/userid`
>
> If you do not specify a data source type and parameter name, an Oracle database is assumed.

3. The URL redirects the user to either `rwservlet` or the JSP depending upon whether this report has been set to execute via `rwservlet` or a JSP.

4. If you used `SSOCONN` in your URL, `rwservlet` checks the Single Sign-On key against the Oracle Internet Directory to see if it already has been mapped to a data source connection string (e.g., `scott/tiger@my_or_db`).

5. If you used `SSOCONN` and the Oracle Internet Directory already has a connection string associated with the key, then `rwservlet` uses that connection string for the data source connection of the report. If you used `SSOCONN` but Oracle Internet Directory does not already contain a connection string for the key, the Oracle Delegated Administration Services Create Resource page is displayed and the user must enter their data source connection string. See Figure 9–1. Oracle Delegated Administration Services stores the

string in Oracle Internet Directory for future use and `rwservlet` uses the newly entered connection string for the data source connection string of the report.

> **Note:** Because of this feature, many users can use the same report URL even if they all use different data source connection strings.

Figure 9–1   Oracle Delegated Administration Services Create Resource
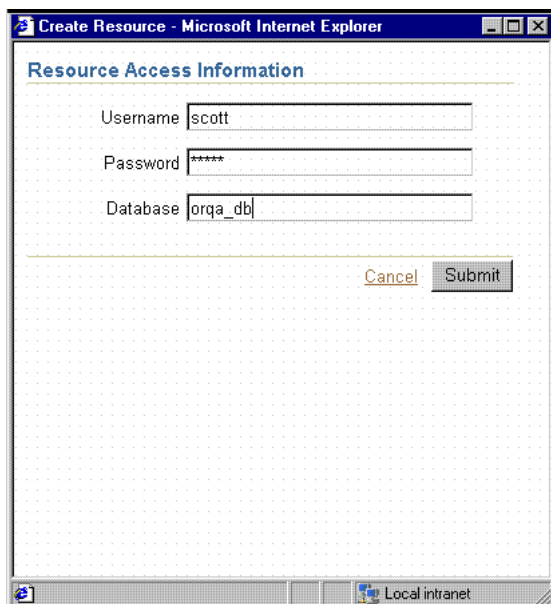
6. `rwservlet` constructs a command line from the URL (and Oracle Internet Directory information if you used `SSOCONN`) and passes it to the Reports Server.

7. The Reports Server validates the user credentials against the Oracle Internet Directory server. The Reports Server checks whether the user has the necessary privileges to run the report on the specified server at the specified time to the specified destination. If the validation check fails for any reason, then an error condition is returned to the user and the process terminates.

8. The Reports Server delegates the job to a Reports Engine that accesses the data source, retrieves the data, and formats the report.

9. The completed output is sent to the specified destination. Depending upon the destination, the output may be sent to a printer, served back to the browser, or stored in a file for future reference.

9.1.2.2 Handling Report Requests without OracleAS Single Sign-On

If Single Sign-On is not being used, then any user accessing a secured instance of the Reports Server is challenged to identify themselves by `rwservlet` via its own authentication mechanism (identical to the behavior of Oracle Reports6*i*). Because the HTTP 1.0 protocol is stateless (i.e., each call to the server is effectively independent of all others), users might need to authenticate themselves for each report request unless a cookie is maintained. To allow users to authenticate themselves only once per session, rwservlet has its own client-side cookie, the authid cookie, in which it stores the required authentication information for the current session. Once the user is authenticated, an encrypted cookie is created in the browser to enable the user to submit multiple report jobs without re-authenticating for each request.

> **Note:** If you want to force users to authenticate themselves for a specific report, you can use the SHOWAUTH command line option. Alternatively, you can include a %S in the corresponding report entry in the key map file. This file is usually called cgicmd.dat and is located in *ORACLE_HOME*\reports\conf. %S forces users to enter their username and password each time the report is called.

The `authid` cookies are terminated when the user closes their browser session, but you should not rely strictly on this method of terminating the cookie. You should limit the lifetime of the cookie within a given session. For example, a user might log on and then go to lunch, leaving the browser session open. To minimize the potential for a security breach in this situation, the administrator may specify the COOKIEEXPIRE parameter in the `rwservlet.properties` file. When `rwservlet` receives a job request, 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 (e.g., 30 minutes), the cookie is rejected and the user is challenged to provide authentication information.

> **See Also:** Configuring the Reports Servlet
>
> For more information about the COOKIEEXPIRE parameter and the `rwservlet.properties` file.

9.1.2.2.1   Report Request Flow without OracleAS Single Sign-On In this scenario, the report request is sent to a secured Reports Server with Single Sign-On disabled. In this case, `rwservlet` or a JSP report might be called through the use of a bookmark or from an OracleAS Portal component.

1. The user must somehow gain access to the URL that launches the report request (e.g., via a link on a Web page or a bookmark), and choose the URL.

2. `rwservlet` checks for the AUTHID parameter on the URL or an existing Oracle Reports Authid Cookie. If it finds the AUTHID parameter, it uses that to authenticate the user. If it does not find the AUTHID parameter, it looks for an existing Reports Authid Cookie. (If the report is launched from OracleAS Portal, AUTHID is added to the URL automatically.) If neither AUTHID nor a Reports Authid Cookie is found, rwservlet displays the System Authentication screen, where the user must supply their Single Sign-On username and password. This information is subsequently stored in the Reports Authid Cookie.

3. If only partial data source credentials are provided on the URL (e.g., USERID=scott@orqa), the Database Authentication page is displayed with the partial credentials priming the fields. The user must supply the remainder of the data source credentials before proceeding further. Note that you can control which Database Authentication page is used via the DBAUTH parameter in the `rwservlet.properties` file. If no data source credentials at all are provided, the Database Authentication page is not displayed and it is assumed the report does not require a data source.

   > **See Also:**   Configuring the Reports Servlet
   >
   > For more information about the DBAUTH parameter and the `rwservlet.properties` file.

4. `rwservlet` constructs a command line with the necessary information from the previous steps and passes it to the Reports Server.

5. The Reports Server validates the user credentials against the Oracle Internet Directory. It also checks whether the user has the necessary privileges to run the report on the specified server at the specified time to the specified destination. If the validation check fails for any reason, then an error condition is returned to the user and the process terminates.

6. The data source credentials are stored in a Reports Userid Cookie for possible future reference. Note that Pluggable Data Source credentials are not stored in the Reports Userid Cookies.

7. The Reports Server delegates the job to a Reports Engine that accesses the data source, retrieves the data, and formats the report.

8. The completed output is sent to the specified destination. Depending upon the destination, the output may be sent to a printer, served back to the browser, or stored in a file for future reference.

9.1.3 Leveraging Oracle Identity Management Infrastructure

OracleAS Reports Services can take advantage of the capabilities in OracleAS Single Sign-On, which is part of the Oracle Identity Management infrastructure.

9.1.3.1 OracleAS Single Sign-On

With the increasing number of Web-based, e-business applications that companies deploy for use by their employees, customers, and partners, many businesses must now consider Single Sign-On functionality. Single Sign-On refers to the ability to log on to a single security system once, rather than logging on separately to multiple security systems. With Single Sign-On, each user maintains a single identity and password for all data and associated resources to which they need access.

Within a given Web application, OracleAS Reports Services eases the user's experience with OracleAS Single Sign-On. OracleAS Single Sign-On ensures that each user authenticates only once.

9.1.3.1.1 Single Sign-On Components Figure 9–2 provides an overview of the Single Sign-On component architecture.

Figure 9–2   SSO Architecture
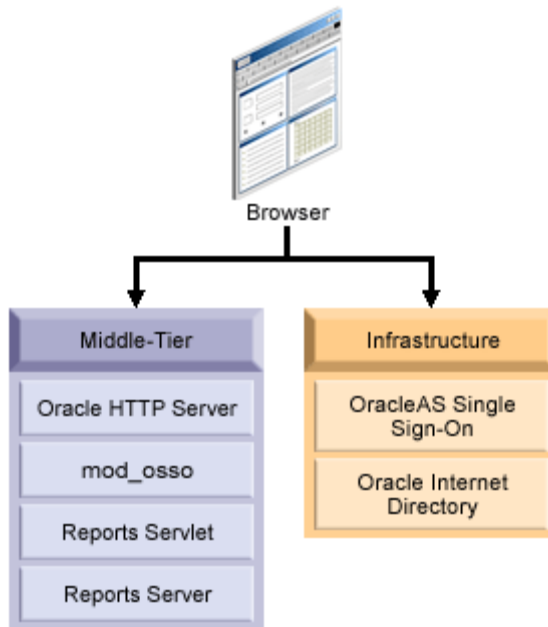
The components of the Single Sign-On environment include:

- A client **Web browser**

- **Oracle HTTP Server**

   The Oracle HTTP Server processes requests from the client browser.

- **Reports Servlet**

- Oracle Application Server Containers for J2EE

   The Reports Servlet is a component of OracleAS Reports Services that runs inside of the Oracle HTTP Server's Oracle Application Server Containers for J2EE (OC4J). When a report request comes to the Oracle HTTP Server, the Reports Servlet passes the job request to the Reports Server.

- **Reports Server**

   The Reports Server processes client requests, which includes ushering them through authentication and authorization checking, scheduling, caching, and distribution.

- **OracleAS Single Sign-On**

  OracleAS Single Sign-On is responsible for managing users' Single Sign-On sessions. It verifies users' login credentials by looking them up in the Oracle Internet Directory.

- **Oracle Internet Directory**

  Oracle Internet Directory is Oracle's highly scalable, native LDAP version 3 service and hosts the Oracle common user identity. OracleAS Single Sign-On authenticates users against the information stored in Oracle Internet Directory. As noted in earlier sections, when Single Sign-On is enabled for OracleAS Reports Services, it checks the Oracle Internet Directory for user and group privilege information. It also retrieves data source connection information from the Oracle Internet Directory.

- **Oracle Delegated Administration Services**

  The Delegated Administration Service provides a comprehensive interface for making updates to the Oracle Internet Directory. OracleAS Reports Services displays Oracle Delegated Administration Services when it encounters a Single Sign-On key that does not already have a data source connection string associated with it in the Oracle Internet Directory.

For more information, refer to Chapter 10, "Configuring and Administering OracleAS Single Sign-On".

9.2 Configuring OracleAS Reports Services Security

This section provides an overview of configuration considerations for OracleAS Reports Services.

9.2.1 Configuring OracleAS Reports Services Security Options

The out-of-the-box implementation of OracleAS Reports Services security includes all of the Oracle components described in Resources Protected pre configured to work with your OracleAS Reports Services installation. If you choose to implement your own security configuration, you can follow the steps in Chapter 10, "Configuring and Administering OracleAS Single Sign-On" and Chapter 11, "Deploying Reports in OracleAS Portal" to use all or only some of these components. For example, you can choose to use OracleAS Single Sign-On without implementing data source security or OracleAS Portal. In another configuration, you might choose to use a different Internet directory to store user and group information. If you prefer to implement none of the above security components,

you can still configure a secured Reports Server, which provides security similar to that available in Oracle Reports6*i*.

> **Note:** At the highest level, all communication to and from Oracle HTTP Server may be configured to use SSL. The Oracle HTTP Server incorporates an OpenSSL module to provide support for Secure Sockets Layer (SSL) and HTTP Secure Sockets Layer (HTTPS). Once this is set up in the Oracle HTTP Server (see *Oracle HTTP Server Administrator's Guide*), `rwservlet` automatically detects the SSL port number.

9.2.1.1 OracleAS Portal

OracleAS Portal provides a number of security features available to OracleAS Reports Services that enable you to ensure that the appropriate users are getting important data in a secure fashion. With OracleAS Portal security features in place, your users see only the data they're supposed to see.

Use OracleAS Portal to control:

- Who has access to each report
- When a report can be run
- Which servers and printers can be used to run a report
- Which report parameters a user can edit with what range of values

OracleAS Portal is a browser-based, data publishing and developing solution that offers Web-based tools for publishing information on the Web and building Web-based, data-driven applications.

OracleAS Portal is tightly integrated with OracleAS Reports Services to create a robust and secure data publishing environment. OracleAS Portal provides easy-to-use wizards for setting up OracleAS Reports Services security. These include wizards for defining user access to reports, Reports Servers, printers, output formats, and report parameters.

Once you define access control information, it's stored in the OracleAS Portal repository. As an OracleAS Portal user, you can then, optionally, publish registered RDFs and JSPs to an OracleAS Portal page. As with all OracleAS Portal functionality, using Portal to deliver your reports is not required. You can deliver reports through command lines, as you may always have, and still benefit from the access control features available to you through OracleAS Portal.

Access to OracleAS Reports Services' security features is not dependent on whether you also use Portal to publish report links or report content. Even if you don't publish via Portal, you can still take advantage of the OracleAS Reports Services' security features available in OracleAS Portal to control user access to all of your reports.

When you expose a report as a portlet through OracleAS Portal, OracleAS Reports Services leverages the Single Sign-On feature. Single Sign-On eliminates the need for users to enter multiple logins, first to the portal then to each of the reports exposed through portlets within the portal. With OracleAS Single Sign-On, when you log in, OracleAS Portal automatically logs you into all registered portlet providers and subsystems.

> **See Also:** OracleAS Portal
>
> For a detailed description of report request flow within OracleAS Portal.

Refer to the *Oracle Application Server 10g Security Guide* for more information about OracleAS Single Sign-On and OracleAS Portal. You'll find this and other related documentation on the Oracle Technology Network, (`http://otn.oracle.com`).

For more information, refer to Chapter 11, "Deploying Reports in OracleAS Portal"

9.2.1.2 Security Interfaces

The Security API of the Reports Software Development Kit (RSDK) allows you to integrate your own security model with the Reports Server. OracleAS Reports Services enables you to plug in any security you wish, using the provided API.

The Security API can control:

- Who has access to each report
- When a report can be run
- Which servers and printers can be used to run a report
- Which report parameters a user can edit with what range of values

The RSDK includes a tutorial that shows you how to integrate your own security using an XML file to store the authorization information. At the end of this tutorial, you will be able to:

- Implement a security class with OracleAS Reports Services
- Register a security class with OracleAS Reports Services

- Use the security class with OracleAS Reports Services

The tutorial and further information on the RSDK can be found on the Oracle Technology Network.

10

Configuring and Administering OracleAS Single Sign-On

Single Sign-On enables you to establish a unique identity for each user, and tie that identity to the resources and data sources unique to that user. For example, a user might log into an environment such as OracleAS Portal, which enables them to access certain reports and printers for which they have the necessary privileges. When they choose to run a report from this environment, they can access the necessary data sources for the report because their data source credentials are stored with the single user identity used to login to OracleAS Portal. Thus, logging in once provides them access to all of the resources and data sources they require to run their reports.

Because OracleAS Reports Services provides a flexible approach to security, you can implement many variations of the configuration described above. For example, you might choose not to store data source credentials with the single user identity. Or you might prefer to use direct URLs for launching reports rather than a platform like OracleAS Portal. If your reports are public and do not require any security, then you might choose to turn off report security altogether.

This chapter describes how you can implement and administer various configurations of OracleAS Single Sign-On with OracleAS Reports Services.

- Prerequisites
- Configuring Out-of-the-Box OracleAS Single Sign-On
- Administering OracleAS Single Sign-On
- Choosing the connecting entity for the Oracle Internet Directory
- OracleAS Forms Services Security Considerations

10.1 Prerequisites

OracleAS Single Sign-On can be implemented only in a secure server environment. This means that you must have a security policy in place in your Reports Server configuration file before you can consider implementing OracleAS Single Sign-On with OracleAS Reports Services.

> **Note:** Security settings are discussed in the following places: Chapter 3, "Configuring OracleAS Reports Services" tells you how to specify the Java class that defines the security policy for the server; Chapter 11, "Deploying Reports in OracleAS Portal" tells you how to deploy OracleAS Reports Services reports in OracleAS Portal; Appendix A, "Command Line Options" provides information about the SSOCONN command line option.

With OracleAS Single Sign-On, your administrator establishes a user identity for each user. The administrator does this in the Oracle Internet Directory, through its user interface, the Oracle Delegated Administration Services. You can access Oracle Delegated Administration Services standalone or through OracleAS Portal. In either case, the information is saved to the Oracle Internet Directory.

The user identity is comprised of the user name and password. Once users are established, data source connection strings may be associated with them. At login, users must enter their user names and passwords (their user identities), which will in turn give them access to all of the data sources associated with those identities. OracleAS Single Sign-On issues a session cookie that effectively acts as a key that opens all authorized doorways for that session.

> **Note:** For detailed information about the requirements and procedures required for setting up SSO-related components, such as the Oracle Internet Directory, see the *Oracle Internet Directory Administrator's Guide* and the *Oracle HTTP Server Administrator's Guide* on the Oracle Application Server documentation CD and on the Oracle Technology Network, (`http://otn.oracle.com`).

10.2 Configuring Out-of-the-Box OracleAS Single Sign-On

By default, the Reports Server is secured and, to run a report, you must login with a valid Single Sign-On userid and password. The Reports Server is configured by default with the OracleAS Single Sign-On instance installed as part of Oracle

Application Server. The Oracle Internet Directory instance installed with Oracle Application Server is used as the default repository for user and group information. If you want to configure the Reports Server to use a different Oracle Internet Directory instance or disable security, refer to Administering OracleAS Single Sign-On. For information on how to add users to the Oracle Internet Directory, refer the *Oracle Internet Directory Administrator's Guide*. In addition, for each Oracle Application Server installation, the Reports Server instances connect to the Oracle Internet Directory as an application entity that is unique to the Oracle Application Server installation. For more information on this behavior, refer to Connecting to the Oracle Internet Directory.

If a user is not already logged in to OracleAS Single Sign-On, they are prompted to login when they attempt to run a report to the Reports Server via `rwservlet`. If the user parameters for a report include SSOCONN, OracleAS Single Sign-On will search for the user's data source credentials in the Oracle Internet Directory. If none are found, then OracleAS Single Sign-On prompts the user to create a new resource. For more information on rwservlet, refer to `rwservlet`. For more information about SSOCONN, refer to SSOCONN.

The Reports Server is also configured to operate with OracleAS Portal by default. You can optionally add reports to the portal and enable users to launch them from the portal. Since users must login to the portal in this case, they are not prompted to login again when they launch their reports because they have already been identified to OracleAS Single Sign-On by logging in to the portal.

You can also optionally define access controls for resources associated with the Reports Server (e.g., reports, printers, Reports Servers, and calendars) in OracleAS Portal. To control access to resources, you must add them to the portal and specify their access options. The resource access controls you specify in OracleAS Portal apply to reports that you run outside of the portal as well. For example, if a user tries to run a report through rwservlet, it will be subject to any access controls you have put in place through OracleAS Portal.

> **See Also:** Deploying Reports in OracleAS Portal
>
> For more information about the integration between OracleAS Portal and OracleAS Reports Services.

10.3 Administering OracleAS Single Sign-On

This section describes some of the administrative tasks you may need to perform as you maintain security for OracleAS Reports Services.

■ Enabling and Disabling OracleAS Single Sign-On

- Enabling and Disabling Reports Server Security
- Enabling and Disabling Data Source Security
- Connecting to the Oracle Internet Directory

10.3.1 Enabling and Disabling OracleAS Single Sign-On

To take advantage of OracleAS Single Sign-On out-of-the-box, the SINGLESIGNON parameter in the rwservlet configuration file (rwservlet.properties) is set to YES, which indicates that you will use OracleAS Single Sign-On to authenticate users. You may change this parameter to NO, if you choose not to use OracleAS Single Sign-On. If you choose NO, the Reports Server authenticates users by itself (i.e., the same mechanism used in Oracle Reports6*i*). The rwservlet configuration file is usually found in:

```
ORACLE_HOME\reports\conf
```

The SINGLESIGNON value is usually commented out after installation, but the default value is YES.

> **Note:** OracleAS Reports Services is configured for OracleAS Single Sign-On out-of-the-box. Oracle considers this to be the normal security deployment model and you should only turn it off if you plan to run in a completely custom security configuration.

10.3.2 Enabling and Disabling Reports Server Security

Reports Server security is turned on and off in the Reports Server configuration file. By default, the Reports Server configuration file, *ORACLE_HOME*/reports/conf/*servername*.conf, contains a security element like the following:

```
<security id="rwSec" class="oracle.reports.server.RWSecurity">
    <!--property name="securityUserid" value="portal_id/portal_password@portal_
    schema" confidential="yes" encrypted="no"-->
    <property name="oidEntity" value="%REPORTS_OID_ENTITY%" confidential="yes"
    encrypted="no"/>
</security>
```

> **Note:** In releases prior to 10*g* (9.0.4), the `securityUserid` property was specified differently. In 10*g* (9.0.4) and later releases, the old property specification is still provided but commented out.

This security element is referenced by default from the two default job elements in the configuration file to indicate that Reports Server security should be enforced:

```
<job jobType="report" engineId="rwEng" securityId="rwSec"/>
<job jobType="report" engineId="rwEngURL" securityId="rwSec"/>
```

To disable Reports Server security, you must remove or comment the `security` element as well as the `secuirtyId` attributes from the `job` element specifications.

10.3.3 Enabling and Disabling Data Source Security

To enable data source security through OracleAS Single Sign-On, you must do the following:

- Include `SSOCONN` in the URL that launches the report.
- Populate the Oracle Internet Directory with data source connection information using one of three methods.

If you wish to implement data source security through OracleAS Single Sign-On for your own pluggable data sources, you need to perform the following additional task:

- Add a new resource type to the Oracle Internet Directory

The sections that follow explain how to perform these operations.

10.3.3.1 SSOCONN

To enable data source security through OracleAS Single Sign-On, the URL must contain or reference (i.e., via the key map file) a OracleAS Single Sign-On parameter (`SSOCONN`) with a value of the form:

key_name/data_source_type/parameter_name

`key_name` maps to a string stored in the Oracle Internet Directory that provides the necessary information to connect to the database. When Oracle Reports encounters a `key_name`, it checks to see if the current user has a corresponding key stored in the Oracle Internet Directory. If so, Oracle Reports uses the string stored in that key to connect to the data source. If not, Oracle Reports checks to see if the `key_name`

maps to a publicly available key. If so, Oracle Reports uses that key. If not, Oracle Delegated Administration Services prompts the user to create a new resource.

> **See Also:** Populating the Oracle Internet Directory
>
> For more information about populating the Oracle Internet Directory with resources

`data_source_type` is the kind of data source to which you are connecting. `data_source_type` lets Oracle Reports know what format to expect in the string associated with `key_name`. `data_source_type` must be a valid resource type stored in the Oracle Internet Directory. Oracle Reports provides default resource types for the following:

- Oracle database (OracleDB)
- JDBC (jdbcpds)
- Oracle Express (EXPRESSPDS)

You can also create addition resource types in the Oracle Internet Directory for your own pluggable data sources.

> **See Also:** Adding a New Resource Type
>
> For more information about adding resource types.

`parameter_name` specifies the Oracle Reports system or user parameter to be used to pass the connection string to Oracle Reports. For example, in the case of the OracleDB data source, Oracle Reports receives the connection string via the `USERID` parameter and uses it to connect to the specified Oracle database. Similarly, for the EXPRESSPDS, the `EXPRESS_SERVER` parameter is used and, for the jdbcpds, `p_jdbcpds` is used. If you have your own custom pluggable data sources, you would need to define your own user parameter for passing the connection string to Oracle Reports and specify it as `parameter_name` for `SSOCONN`.

10.3.3.1.1 Oracle Database Example In the case of an Oracle database, the URL to call a report with `SSOCONN` would look something like the following:

```
http://myhost.mycompany.com:7779/reports/rwservlet?server=rs_cped
&report=my.rdf&destype=cache&ssoconn=mykey/OracleDB/userid&desformat=html
```

10.3.3.1.2 Oracle Express Example In the case of an Oracle Express database, the Single Sign-On value would look something like the following:

```
http://myhost.mycompany.com:7779/reports/rwservlet?server=rs_cped
```

```
&report=exppds.rdf&destype=cache&ssoconn=exptest1/EXPRESSPDS/express_server&desformat=html
```

10.3.3.1.3 JDBC Pluggable Data Source Example In the case of a JDBC data source, the Single Sign-On value would look something like the following:

```
http://myhost.mycompany.com:7779/reports/rwservlet?server=rs_cped
&report=Jdbcthin.rdf&destype=cache&desformat=html&ssoconn=jd1/jdbcpds/p_jdbcpds
```

In this case, jd1 is an Oracle Internet Directory resource name.

> **See Also:**   Configuring and Using the JDBC PDS
>
> For more information on how to configure a JDBC data source.

Usage Notes

- When you use SSOCONN in a command line, you cannot:

    - Specify authid in the same command line.

    - Run against a Reports Server that is not secure.

    - Have SINGLESIGNON set to NO in rwservlet.properties.

    Doing any of the above with SSOCONN in the command line results in an error.

10.3.3.2 Populating the Oracle Internet Directory

For data source security to function with OracleAS Single Sign-On, you need to store the data connection information for each user in the Oracle Internet Directory or make the resource a default one available to every user. You can populate Oracle Internet Directory with this information in any one of the following ways:

- Oracle Delegated Administration Services

- User Prompt

- Batch Loading

- Making a Resource Available to All Users

10.3.3.2.1 Oracle Delegated Administration Services If you only need to enter credentials for a small number of users (e.g., for a development environment), you can use Oracle Delegated Administration Services to directly enter connection string information into the Oracle Internet Directory for each user.

> **Note:** Before a user can access Oracle Delegated Administration Services, an administrator must have already entered a user identity in the Oracle Internet Directory for the user. This step can be done by batch loading information that is already entered into an LDAP directory in some other source.
>
> See the *Oracle Internet Directory Administrator's Guide* for information on batch loading. You'll find it on the Oracle Application Server documentation CD and on the Oracle Technology Network, (`http://otn.oracle.com`).

During Oracle Application Server installation, you specify the location of Oracle Delegated Administration Services. You use this URL to access Oracle Delegated Administration Services for administrative purposes. Once in Oracle Delegated Administration Services, you enter the information via the Resource Access Information section of the Preferences tab for the user. See Figure 10–1. Note that, for the Preferences tab to appear, there must already be a resource in place.

Figure 10–1   Delegated Administration Services Preferences
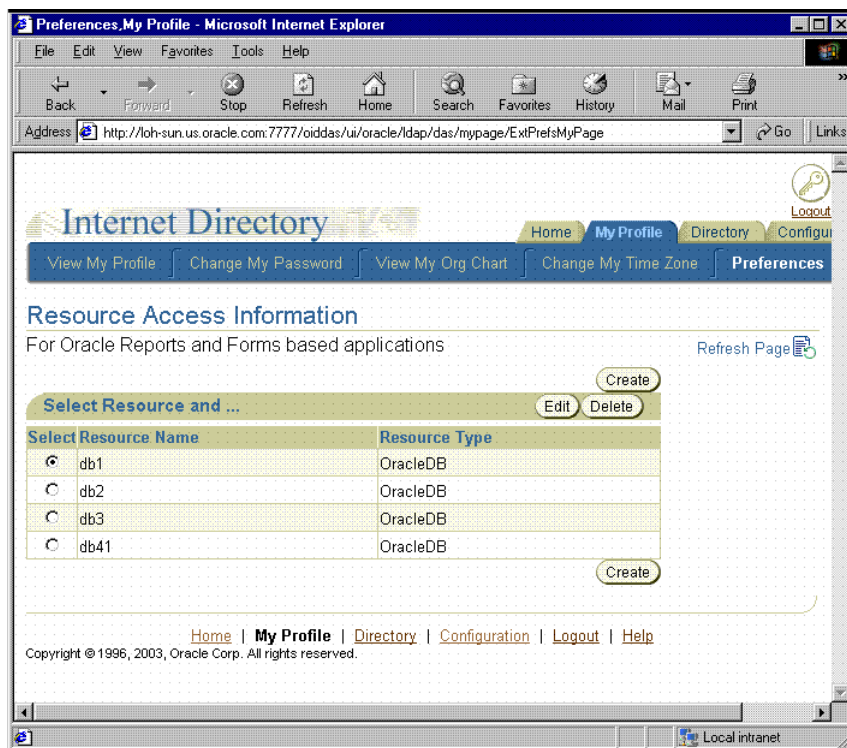
If you need to enter data source information for a large number of users, you should use either the user prompt or batch methods of populating the Oracle Internet Directory.

10.3.3.2.2   User Prompt   If you prefer to have users enter their own connection string information, you do not have to pre-populate the Oracle Internet Directory with data source connection information at all. If you use SSOCONN when launching the report but the Oracle Internet Directory does not already contain a connection string for the key and the key is not publicly available to all users, the Oracle Delegated Administration Services Create Resource page is displayed to the user, who must enter their data source connection string. See Figure 10–2. Oracle Delegated Administration Services stores the string entered by the user in the Oracle Internet Directory for future use and rwservlet uses the newly entered connection string for the data source connection string of the report.

> **Note:** Because of this feature, many users can use the same report URL even if they all use different data source connection strings.

Figure 10–2   Oracle Delegated Administration Services Create Resource
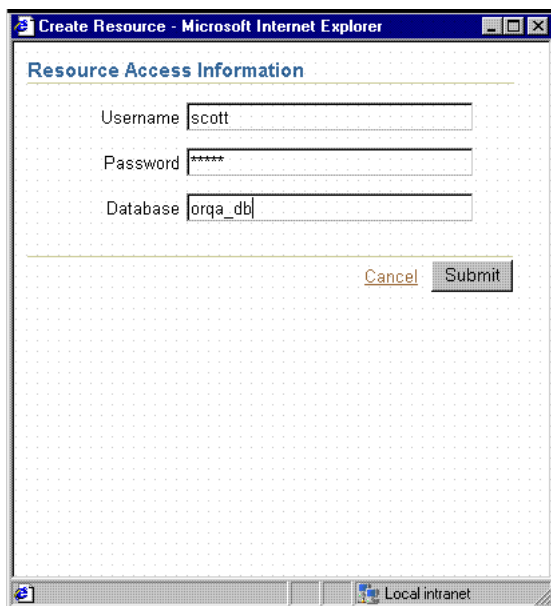
> **Note:** In the Create Resource dialog, if you want to enter a JDBC connection string, you can do so by entering `hostname:port:sid` in the Database field.

10.3.3.2.3   Batch Loading   Resources for OracleAS Reports Services are created in the Oracle Internet Directory under the following entry:

```
orclresourcename=resource_name, cn=Resource Access Descriptor,
orclownerguid=guid, cn=Extended Properties, cn=OracleContext,
dc=us,dc=oracle,dc=com[1]
```

---

[1]   dc=us,dc=oracle,dc=com is merely an example in this instance. You would normally enter your own values for these items.

Before You Begin   You need to create orclownerguid=*guid* in the above Oracle Internet Directory entry before you can proceed with the batch loading of resources. If you used Oracle Delegated Administration Services to create your users, orclownerguid=*guid* was created automatically and you can proceed to Batch Loading Resources.

If you seeded users into the Oracle Internet Directory with an LDIF file, then, before following the steps in Batch Loading Resources, you need to complete the following steps:

1. Get the users' GUIDs.

Depending on how your users are created in the Oracle Internet Directory, you can use any number of methods to get their GUIDs. You can get user GUIDs using the Oracle Internet Directory LDAP API. You can also get it using the ldapsearch command:

```
D:\ora9ias\BIN>ldapsearch -h host_name -p port_num -L -D cn=orcladmin
-w orcladmin's_password -b "cn=users,dc=us,dc=oracle,dc=com" -s sub
"objectclass=*" dn orclguid
```

2. Create the user entry orclownerguid=*guid* under cn=Extended Properties, cn=OracleContext, dc=us, dc=oracle, dc=com.

   a. Modify the sample script, *ORACLE_ HOME*\reports\samples\scripts\createuser.ldif by replacing the place holder with real values.

   b. Load createuser.ldif using ldapadd. For example:

   ```
   D:\ora9ias\BIN>ldapadd -D cn=orcladmin -w welcome1
   -h host_name -p port_num -f createuser.ldif
   ```

3. Once you have created orclownerguid=*guid*, proceed to Batch Loading Resources.

Batch Loading Resources   Follow the steps below to batch load data source resources for your users:

1. Create the user's resource entry orclresourcename=*resource_name*, cn=Resource Access Descriptor under orclownerguid=*guid*, cn=Extended Properties, cn=OracleContext, dc=us, dc=oracle, dc=com, where orclownerguid=*guid* is the GUID created in Before You Begin.

    a. Modify the sample script, *ORACLE_*
HOME\reports\samples\scripts\createresource.ldif by
replacing the place holder with real values.

    b. Load createresource.ldif using ldapadd. For example:

```
D:\ora9ias\BIN>ldapadd -D cn=orcladmin -w orcladmin's_password -h host_
name -p port_num -f createresource.ldif
```

10.3.3.2.4 Making a Resource Available to All Users If you want to make a resource
publicly available to all of your users, you can do so by following these steps:

1. Launch Oracle Delegated Administration Services and go to the **Home** tab.

2. Login as the administrator (orcladmin).

3. Click the **Configuration** tab.

4. Click the **Preferences** sub tab and you should see a page similar to the one in
Figure 10–3.

Figure 10–3   Oracle Internet Directory Configuration Preferences Page
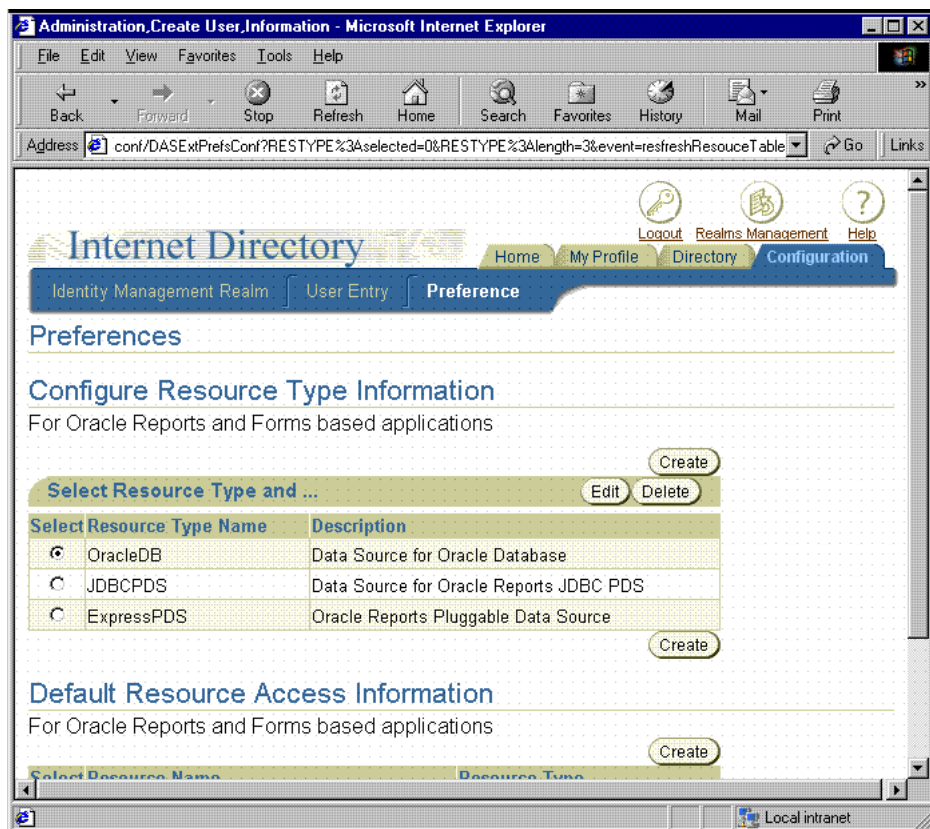
5. Under Default Resource Access Information, click **Create**.

6. In the Create Resource page, enter the resource name and select the Resource type from the drop-down list. For example, JDBCPDS.

7. Click **Next**.

8. Enter the connection information. For example, `scott/tiger@mydb`.

9. Click **Submit**.

10. Click **OK**.

That resource should now appear under Default Resource Access Information and be available to all users.

10.3.3.3 Adding a New Resource Type

If you want to add a new resource type to support your own pluggable data source, you need to perform the following procedure:

1. Launch Oracle Delegated Administration Services and go to the **Home** tab.

2. Login as the administrator (orcladmin).

3. Click the **Configuration** tab.

4. Click the **Preferences** sub tab and you should see a page similar to the one in Figure 10–3.

5. Under Configure Resource Type Information, click **Create** and you should see a page similar to the one in Figure 10–4.

Figure 10–4   Create Resource Type page
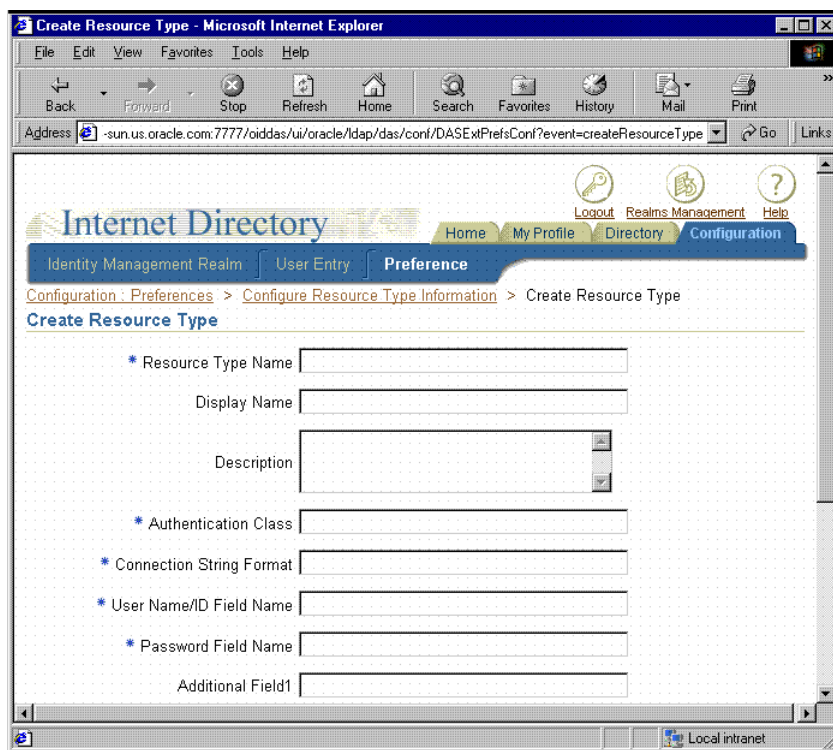
6. Fill in at least the required fields. Field descriptions are provided in Table 10–1.

Table 10–1   Create Resource Type Properties

Property	Description
Resource Type Name	Is the name of the new resource type. This name is used when you need to reference the resource type, for example, in the data_source_type portion of the SSOCONN string.
Display Name	Is the name to be used when the resource type appears in the user interface.
Description	Is a textual description that explains the purpose of the resource type and any other documentary information you want to enter for it.

Table 10–1   Create Resource Type Properties

Property	Description
Authentication Class	Leave this field blank.
Connection String Format	Defines how OracleAS Reports Services should construct the connection string using the values stored in Oracle Internet Directory for the resource. For example:
	for the Oracle database or a JDBC data source your connection string format might be:
	`orclUserIDAttribute/orclPasswordAttribute` `@orclFlexAttribute1`
	This string indicates that the user name is followed by a slash, the password, an at sign (@), and then additional attribute 1 (e.g., for the TNS name of the database). A connection string that adheres to this format would look similar to this one:
	`scott/tiger@db1`
	for Oracle Express your connection string format might be:

	`server=orclFlexAttribute1/domain=orclFlexAttri`
	`bute2/user=orclUserIDAttribute/password=orclPa`
	`sswordAttribute`

	This string indicates that `server=` is followed by the first additional attribute, a slash, `domain=`, the second additional attribute, a slash, the user name, a slash, and the password. A connection string that adheres to this format would look similar to this one:
	`server=a1/domain=a2/user=scott/password=tiger`
User Name/ID Field Name	Is the display name of the user name field that contains the value for `orclUserIDAttribute`. The display name appears on the Create Resource page (Figure 10–2) next to the field for `orclUserIDAttribute`. Typically, you would enter something like Username or User Name for this display name.
Password Field Name	Is the display name of the password field that contains the value for `orclPasswordAttribute`. The display name appears on the Create Resource page (Figure 10–2) next to the field for `orclPasswordAttribute`. Typically, you would enter something like Password or password for this display name.

Table 10–1 Create Resource Type Properties

Property	Description
Additional Field 1-3	Is the display name of the additional fields, which contain the values of `orclFlexAttribute1`, `orclFlexAttribute2`, and `orclFlexAttribute3`. You need to specify these fields for whatever values your connection string requires beyond user name and password. For example, you might use one of them to contain a server or domain name.The display name appears on the Create Resource page (Figure 10–2) next to the field for `orclFlexAttribute1`, `orclFlexAttribute2`, or `orclFlexAttribute3`. Typically, you would enter something descriptive of the field's contents, such as Server or Domain, for this display name.

7. Click **Submit**. Your resource type is created and you can now reference it in the `data_source_type` portion of the SSOCONN argument.

> **See Also:** SSOCONN
>
> For more information about SSOCONN.

10.3.4 Connecting to the Oracle Internet Directory

As you may recall from Chapter 9, "Securing OracleAS Reports Services", OracleAS Reports Services must connect to the Oracle Internet Directory to verify user privileges and obtain existing data source connection information. In connecting to the Oracle Internet Directory, you must consider:

- Choosing the Connecting Entity for the Oracle Internet Directory
- Choosing the Oracle Internet Directory Instance

10.3.4.1 Choosing the Connecting Entity for the Oracle Internet Directory

When OracleAS Reports Services connects to the Oracle Internet Directory, it does so as an application entity. By default, each OracleAS Reports Services application entity is unique to its Oracle Application Server installation. Every Reports Server started from the same Oracle Application Server installation (i.e., *ORACLE_HOME*) uses the same application entity to connect the Oracle Internet Directory. This setup ensures that each Reports Server can only access information in the Oracle Internet Directory that is relevant to its instance of Oracle Application Server.

For example, suppose you have two instances of Oracle Application Server, one for your Finance group and one for your Human Resources group. A Reports Server

from the Finance group's Oracle Application Server instance would be prevented from accessing information relevant only to the Human Resources group, and vice versa. Thus, information stored in the Oracle Internet Directory is more secure by default.

In previous releases of OracleAS Reports Services, all Reports Servers connected to the Oracle Internet Directory as the same application entity. As a result, it was not possible to restrict a Reports Server's access to information in the Oracle Internet Directory.

To revert to the less restrictive security mode, refer to the OracleAS Reports Services chapter of the Oracle Application Server Release Notes.

10.3.4.2 Choosing the Oracle Internet Directory Instance

By default, the Reports Server is configured to use the Oracle Internet Directory instance installed with Oracle Application Server. If you are building your system anew, this arrangement is fine. If, however, you have an existing Oracle Internet Directory instance that you want to use for the Reports Server, you have to make some adjustments to your configuration.

Changing Oracle Internet Directory instances, though, must be done as part of a complete change of your Oracle Application Server middle tier. For more information about this process, please refer to the chapter on reconfiguring application server instances in the *Oracle Application Server 10g Administrator's Guide*.

10.4 Choosing the connecting entity for the Oracle Internet Directory

You can merge serveral application entities so that the Reports Servers installed in separate *ORACLE_HOME*s can share available SSOCONN resources. To achieve this merge, you must execute an `ldif` file with the `ldapmodify` command. The `ldif` file should contain the following:

```
dn: dn of the group which represents the logical grouping of all report
instances
changetype: modify
add: uniquemember
uniquemember: dn of the Reports Application Entity
```

where:

```
dn of the group which represents the logical grouping of all report
instances=cn=Virtual Application Group, orclApplicationCommonName=reports_
application_entity_name², cn=Reports, cn=Products, cn=OracleContext
```

and

```
dn of the Reports Application Entity =orclApplicationCommonName=reports_
application_entity_name, cn=Reports, cn=Products, cn=OracleContext
```

10.5 OracleAS Forms Services Security Considerations

The default configuration for Oracle Application Server Forms Services does not run in OracleAS Single Sign-On (SSO) mode. The default configuration for OracleAS Reports Services does run in SSO mode.

Forms applications calling integrated OracleAS Reports Services using the RUN_ REPORT_OBJECT built-in will not experience any problems when OracleAS Forms Services is running in non-SSO mode and OracleAS Reports Services is running in SSO mode as long as the Reports Server and the requested report are not registered in OracleAS Portal.

Other Requirements:

- The property, Reports Server, must be set explicitly for all report objects in the Oracle Forms module.

- If a Reports Server other than the default is being used, that server must be started from the command line as follows:

  ```
  rwserver server=Reports Server name
  ```

- The system variable, REPORTS_PATH, must be modified in the file *ORACLE_ HOME*/bin/reports.sh to reference the path of the reports to be run.

- The first time a Reports Server is started, it creates a configuration file called *server_name*.conf located in the *ORACLE_HOME*/server/conf/ directory.

- The default status of a Reports Server is secure. To change the Reports Server status to non-secure, modify *ORACLE_HOME*/server/conf/*reports_ server_name*.conf by commenting out the <security> tag and removing securityId from the <job> tags.

- After making these modifications, the Reports Server must be stopped and restarted.

---

[2] Where reports_application_entity_name is in the format, reportsApp_hostname_GUID. For example, reportsApp_serv1.us.oracle.com_ C7543D42A9E26726E034080020A46EE2

- If OracleAS Forms Services is configured to run in SSO mode, then report requests are sent with the authid provided, based on the SSO user login.

- Protected reports and Reports Servers can be registered in OracleAS Portal.

Table 10–2 lists the possible Forms/Reports combinations and expected results:

Table 10–2   Outcome of Forms/ Reports Integration when Forms is running in SSO Mode or Non-SSO Mode

Report Type	Registered, Secure Reports Server (runs only registered reports)	Registered, Secure Reports Server (runs any reports)	Non-Secure Reports Server
Reports with public access	report generated	report generated	report generated
Reports with specific user access	report generated	report generated	report generated
Reports with no specific user access	report not generated	report not generated	report generated
Non-registered reports	report not generated	report not generated	report generated

11

Deploying Reports in OracleAS Portal

This chapter describes how to use OracleAS Portal to deploy your Oracle Application Server Reports Services reports. It includes the following sections:

- Creating Reports Users and Named Groups in OracleAS Portal
- Registering Oracle Reports Components in OracleAS Portal

Before you deploy reports, both OracleAS Portal and Oracle Application Server Reports Services must be installed and configured.

> **See also:** The following resources for further information:
>
> - Chapter 3, "Configuring OracleAS Reports Services" for information on configuring Oracle Application Server Reports Services
> - the *Oracle Application Server Portal Configuration Guide* for information on configuring OracleAS Portal
> - the *Oracle Application Server 10g Installation Guide* for information on installing both components
> - the Oracle Application Server documentation CD
> - the Oracle Technology Network, (`http://otn.oracle.com`)

11.1 Creating Reports Users and Named Groups in OracleAS Portal

If you use the security features in OracleAS Portal to control access to your reports, you must register all of your Reports users in the Oracle Internet Directory (OID) and assign security privileges to all of them through OracleAS Portal.

> **Note:** If you have a large user population already entered into an LDAP-compatible directory, you can use Oracle Internet Directory (OID) features to synchronize the directories and save yourself the effort of entering your users individually. You'll find information about OID's Directory Integration Server in the *OID Administrator's Guide*.

In OracleAS Portal, security privileges can be granted to individual users and to named groups of users. Named groups are useful for streamlining the process of granting access privileges. You can assign a set of access privileges to a named group, and grant the entire set of privileges to an individual simply by adding that person to the group.

> **Note:** When you use features like OracleAS Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by Oracle Application Server Reports Services may override the initial NLS_LANG setting. This change may in turn affect the behavior of the running report, such as bidirectional output in PDF. On UNIX platforms, you can work around this issue by setting the NLS_LANG explicitly in report.sh. You can also use the new environment switching functionality to dynamically set the environment for reports. Refer to Dynamic Environment Switching, for more information.

The next sections provide overview information on how to create users and groups in OracleAS Portal. They include:

- Default Reports-related Groups
- Creating Users and Groups

11.1.1 Default Reports-related Groups

When you install OracleAS Portal, Reports-related groups are created for you automatically. These include the following groups:

- RW_BASIC_USER
- RW_POWER_USER
- RW_DEVELOPER

- RW_ADMINISTRATOR

You need to assign appropriate privileges to these groups to enable group members to perform any desired functions on reports through OracleAS Portal. For example, for each report object that you want members of a group (e.g., RW_BASIC_USER) to be able to run, you have to grant the Execute privilege to that group from the Access tab of the report object. Similarly, if you want members of a group (e.g., RW_ADMINISTRATOR) to be able manage Reports Servers, printers, and reports, you have to grant the Manage privilege to that group from the Access tab of those objects.

While you can assign object privileges to individual users, we recommend that every person who will access your reports belong to one of these groups or a group that you create yourself. If users try to run reports without being a member of one of these groups, by default, they are assigned the privileges of a basic user.

11.1.1.1 RW_BASIC_USER

Should the security check fail, the users in RW_BASIC_USER see less detailed error messages than the users in other Oracle Reports groups, such as:

```
Security Check Error
```

Typically, you will want to assign this group minimal privileges. For example, you probably will want to give RW_BASIC_USER the privilege to execute reports and no more.

11.1.1.2 RW_POWER_USER

In addition to the privileges of the RW_BASIC_USER group, the RW_POWER_USER group sees error messages that are more detailed than those displayed to basic users. For example, if they are not permitted to run to HTML, but they try anyway, they might get the message:

```
Cannot run report to HTML
```

This is more detailed than the message an RW_BASIC_USER would receive for the same error.

11.1.1.3 RW_DEVELOPER

In addition to the privileges of the RW_POWER_USER groups, the RW_DEVELOPER group can run Web commands, such as SHOWENV and SHOWMAP, which show the system environment.

Typically, you would assign privileges to this group needed by a developer who is testing reports. Depending upon your installation, you might even assign them limited administrative privileges.

11.1.1.4 RW_ADMINISTRATOR

In addition to the privileges of RW_DEVELOPER, these users also have access to the administrator's functionality in the Oracle Reports Queue Manager, which means they can manage the server queue, including rescheduling, deleting, reordering jobs in the server, and shutting down a server. RW_ADMINISTRATOR also has the privilege to run Web commands through `rwservlet`.

Typically, you will want to assign to this group some (but probably not all) of the same privileges assigned to the PORTAL_ADMINISTRATORS group.

Note: Initially, only members of the PORTAL_ADMINISTRATORS group have MANAGE privileges for Oracle Reports objects. They can CREATE, UPDATE, and DELETE the registered report definition files, servers, and printer objects in OracleAS Portal. In addition to all the links activated for the developer user, administrators can navigate to the Access tab on the Component Management Page, accessible in OracleAS Portal. This is where the administrator can specify who will have access to this report. People with administrator privileges can assign security privileges for other people and receive full error messages from Oracle Application Server Reports Services.

11.1.2 Creating Users and Groups

OracleAS Portal uses the Delegated Administration Service (DAS) interface to the Oracle Internet Directory (OID) to register users for access to Portal. You can enter the DAS interface through Portal to create new users. The creation of new users and groups is discussed in the *Oracle Application Server Portal Configuration Guide* available on the Oracle Application Server documentation CD.

When you create groups, you need to assign appropriate privileges to them to enable group members to perform any desired functions on reports through OracleAS Portal. For example, for each report object that you want members of a group (e.g., RW_BASIC_USER) to be able to run, you have to grant the Execute privilege to that group from the Access tab of the report object. Similarly, if you want members of a group (e.g., RW_ADMINISTRATOR) to be able manage Reports

Servers, printers, calendars, and reports, you have to grant the Manage privilege to that group from the Access tab of those objects.

Ideally, you should provide a user with the necessary privileges on objects by assigning them to a group that has appropriate privileges for their role. For example, if you are creating a user who needs to be able to run but not manage reports, you could assign her to RW_BASIC_USER. If need be, you may assign object privileges to individual users (e.g., JSMITH) rather than groups, but this approach is more difficult and time consuming to manage.

11.2 Registering Oracle Reports Components in OracleAS Portal

Before you begin, you must have a sufficient level of privileges in OracleAS Portal to access the portlets and complete the tasks required for setting access controls. In order to manage reports in OracleAS Portal, you must belong to both the PORTAL_ADMINISTRATORS and RW_ADMINISTRATOR groups. If you only belong to RW_ADMINISTRATOR, you will encounter errors when you attempt to create report objects.

For more information about joining privilege groups in OracleAS Portal, refer to the *Oracle Application Server Portal Configuration Guide*.

> **See Also:** Running Report Requests
>
> For more information on running reports in OracleAS Portal.

This section outlines the necessary steps to go about:

- Creating an Availability Calendar

- Registering a Printer

- Registering a Reports Server

- Registering a Report

11.2.1 Creating an Availability Calendar

Defining availability calendars is an optional step that allows you to further restrict access to reports, servers, and printers by specifying when they can and cannot be accessed. Availability calendars are not necessary if the reports, the Reports Servers, and printers are always available for processing.

This section provides information on:

- Creating a Simple Availability Calendar

- Creating a Combined Availability Calendar

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

11.2.1.1 Creating a Simple Availability Calendar

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

To create a simple availability calendar:

1. Log in as an administrator to OracleAS Portal.

2. Navigate to the Builder page.

3. Click the **Administer** tab.

4. Click the **Oracle Reports Security Settings** link in the **Oracle Reports Security** portlet.

5. Click the **Create Reports Simple Calendar Access link i**n the Reports Calendar Access portlet on the Oracle Reports Security page.

6. On the resulting page, the **Name** (internal name) and **Portal DB Provider** fields contain default values. To include custom values:

   - Enter a unique name in the **Name** field that will identify the availability calendar internally in OracleAS Portal, for example, MY_CALENDAR. This name must follow the OracleAS Portal rules for a valid component name, i.e.,:

     * It must be no more than 30 characters

     * It must contain only alphanumeric characters (no spaces or special characters allowed).

     * The first character must be a letter (not a number).

   - In the **Display Name** field, enter the name you want to display for this availability calendar when it is exposed through OracleAS Portal. Unlike the internal name, the display name can have spaces in it.

   - Select a **Portal DB Provider** from the provider list of values. All components added to or created in OracleAS Portal must belong to a Portal

DB Provider. This list contains the names of only those providers with which you have privileges to build components.

> **Note:** For information on creating a Portal DB Provider, see the *OracleAS Portal online help*.

7. Click **Next**.

8. Optionally, enter a description of the calendar under **Description**.

9. Click **Next**.

10. On the **Date/Time Availability** page, define the parameters for the calendar:

    Under **Duration**, specify the length of time that comprises a unit of duration (or duration period). For example, if you plan to set this calendar up to allow report access from 9:00 AM to 5:00 PM on a given day, then both **Start** and **End** would be the same month, day, and year, but the hour and minute setting for **Start** would be 9:00 AM and for **End** would be 5:00 PM. In this example, the duration of availability of a report on a given day is from 9:00 AM to 5:00 PM.

    Under **Repeat**, specify how frequently the duration period is repeated:

    - **Occurs only once** means the duration period does not repeat, and associated components are no longer available when the period expires. For example, if you select **Occurs only once** and set a duration period of one year, then the associated components cease to be available after one year.

    - **Yearly** means the duration period restarts each year. If you select **Yearly** and have the same start and end date in your **Duration** setting, but your **Start** hour is set to 9:00 AM and your **End** hour is set to 5:00 PM, then the Reports components associated with this availability calendar will be available one day a year between 9:00 and 5:00.

    - **Monthly** means the duration period restarts each month between the **Start** and **End** dates specified under **Duration**. If you select **Monthly** and have the same date and year in both **Start** and **End**—July 25, 2001—but set the **Start** hour for 9:00 AM and the **End** hour for 5 PM, then the associated components will be available between 9:00 AM and 5:00 PM on the 25th of each month.

    - The **by Date/Day** setting applies only to **Monthly**. With **by Date/Day**, you specify whether the duration period is set by the particular date (e.g., always on the 25th through the 29th of the month) or by the particular

day(s) (e.g., always on Monday through Friday—which happen this month to fall on the 25th through the 29th).

- **Weekly** means the duration period restarts on a weekly basis between the days specified under **Duration**.

- **Daily** means the duration period restarts each day between the hours specified under **Duration**.

- **Frequency** fills in the missing value for the phrase: Repeat every *n* (years, months, weeks, days—depending on what you selected under **Repeat**). For example, if you set the duration period to repeat weekly, then set **Frequency** to 2, the duration period restarts every two weeks, or every other week.

- Optionally, check **Repeat Until** and assign a termination date/time for the calendar. Availability for all associated Reports components ends on the **Repeat Until** date/time.

> **Note:** No validation is run on your calendar. If the duration period exceeds the repetition setting, no error message will be generated. For example, if you set the duration period for 10 days and the repetition for weekly, the periods will overlap, but you will not be notified of the overlap.

11. Click **Next**.

12. On the **Summary** page, click the **Show Calendar** button to preview your availability calendar. If you wish to change some settings, click the **Previous** button and make your changes.

13. On the **Summary** page, click **Finish** to complete the availability calendar.

    The resulting page summarizes your settings for this calendar. On this page, you can edit your settings, get detailed information about the calendar, or delete it.

14. Click **Close** to close this page and return to OracleAS Portal's **Oracle Reports Security** page.

You can combine this calendar with other calendars or apply it "as is" to registered Oracle Application Server Reports Services components.

11.2.1.2 Creating a Combined Availability Calendar

A combined availability calendar combines two or more availability calendars into a single availability calendar. This is useful when you want to set up an availability period, then exclude specific days, such as holidays, from that period.

When you combine calendars, you can indicate that all the days on one of them be excluded from all the days on the other. For example, one calendar could describe availability Monday through Friday; another could describe availability only on Wednesday. You could combine these, excluding the Wednesday calendar, so that the combined calendar describes availability Monday, Tuesday, Thursday, Friday.

Conceivably, you could create a simple calendar that covers the weekdays of an entire year, then multiple additional simple calendars, where one excludes New Years, another excludes a second holiday, another excludes a third, and so on. You could combine all these calendars, excluding all the holiday calendars, so that components were available only on the days your company is open for business, between certain times of day, throughout the year.

To combine availability calendars:

1. Log in as an administrator to OracleAS Portal.

2. Navigate to the Builder page.

3. Click the **Administer** tab.

4. Click **the Oracle Reports Security Settings link i**n the **Oracle Reports Security** portlet.

5. Click the **Create Reports Combined Calendar Access** link in the Reports Calendar Access portlet.

6. Specify an internal name, display name, and Portal DB Provider for the calendar:

   - Enter a unique name in the **Name** field that will identify the combined availability calendar internally in OracleAS Portal, for example, MY_COMBINED_CALENDAR. This name must follow the OracleAS Portal rules for a valid component name, i.e.,:

     * It must be no more than 30 characters

     * It must contain only alphanumeric characters (no spaces or special characters allowed).

     * The first character must be a letter (not a number).

■ Enter the name you want to display for this combined availability calendar in the **Display Name** field. The Display Name is the name that is exposed to your users through OracleAS Portal.

> **Note:** The Display Name, unlike the internal Name, can have spaces in it.

■ Select a **Portal DB Provider** from the provider list of values. All components that you add to or create in Portal must belong to a Portal DB Provider. This list contains the names of only those providers with which you have privileges to build components.

> **Note:** For information on creating a Portal DB Provider, see the *OracleAS Portal online help*.

7. Click **Next**.

8. (Optional) Enter a description of the Availability Calendar in the **Description** field.

9. Click **Next**.

10. On the **Selection** page, highlight the calendars on the **Availability Calendars** list that you want to combine. The calendars are listed by their internal names, not their display names. Use control-click (Windows) or click (UNIX) to select multiple calendars.

    This page lists the availability calendars that have been defined for the same Portal DB Provider under which you are creating this combined availability calendar.

11. Click the right arrow to move the selected calendars to the **Selected Availability Calendars** list.

12. Click **Next**.

13. On the **Exclude** page, highlight the calendar(s) on the **Availability Calendars** list whose dates you want to exclude. Use control-click (Windows) or click (UNIX) to select multiple calendars.

    These are the calendars with dates on which you wish to withdraw availability.

14. Click the right arrow to move the selected calendars to the **Excluded Availability Calendars** list.

15. Click **Next**.

16. On the Summary page, click the **Show Calendar** button to preview your calendar.

    If your exclusion isn't showing up, select a different view. For example, instead of the monthly view, select the weekly.

    If you want to change the combination, close the calendar and click the **Previous** button one or more times to return to the desired page.

17. Click **Finish** to complete creation of the combined calendar.

    The resulting page summarizes your settings for this calendar. On this page, you can edit your settings, get detailed information about the calendar, or delete it.

    > **See Also:** The Manage Portlet
    >
    > For more information on the fields and descriptions listed in the Manage portlet, i.e., Develop, Manage, and Access tabs.

18. Click **Close** to close this page and return to OracleAS Portal's **Oracle Reports Security** page.

You can combine this calendar with other calendars or apply it "as is" to registered Oracle Application Server Reports Services components.

11.2.2 Registering a Printer

It is not required that you register a printer within the security framework of OracleAS Portal. You can run a report on any printer as long as it is available to the Reports Server. However, you might want to confine OracleAS Portal users to a subset of those printers, constrain the use of a printer for certain periods of time, or identify a particular printer to be used for printing output of certain reports.

Printer registration with OracleAS Portal is meaningful for reports that you run through OracleAS Portal as well as those you run through a stand-alone URL.

Once printers are registered within OracleAS Portal, you can associate them with a Reports Server. Many printers can be registered. However, only printers associated with particular Reports Servers are available to print when you register a report with OracleAS Portal and choose those Reports Servers.

You can choose to restrict even further the registered subset of printers that a registered report can be sent to. For example, an Reports Server might be connected to the printer in the office of the CEO, but its selection should not be available to employees running the general ledger report, unless it is the CEO who is running the report. A subset of printers can be listed to the OracleAS Portal user running a report request to select where output should be sent.

Table 11–1   Sample Values

Property	Sample Value
Name (internal name)	`myrep_printer`
Display Name	`My Reports Printer`
Portal DB Provider	`PORTAL_APP`
OS Printer Name	`\\mydomain\printer1`
Availability Calendar	`COMCAL`

To register a printer:

1. Log in as an administrator to OracleAS Portal.

2. Navigate to the **Builder** page.

3. Click the **Administer** tab.

4. Click the **Oracle Reports Security Settings link i**n the **Oracle Reports Security** portlet. The **Oracle Reports Security** portlet enables you to use the security features in OracleAS Portal at the time of defining access to the server, printer, calendar, and reports definition file.

5. Click **the Create Reports Printer Access link i**n the **Reports Printer Access** portlet.

6. On the resulting page, the **Name** (internal name) and **Portal DB Provider** fields contain default values. To include custom values:

   - Enter a unique name in the **Name** field that will identify the printer internally in OracleAS Portal, for example, MY_PRINTER. This name must follow the OracleAS Portal rules for a valid component name, i.e.,:

     * It must be no more than 30 characters

     * It must contain only alphanumeric characters (no spaces or special characters allowed).

* The first character must be a letter (not a number).

■ Enter the name that you want to display for this printer in the **Display Name** field. The Display Name is the name that is exposed to your users through OracleAS Portal.

> **Note:** The Display Name, unlike the internal Name, can have spaces in it.

■ Select the Portal DB Provider that will own the printer from the Portal DB Provider list of values. The Portal DB Providers displayed are those in which you have privileges to build components.

> **Note:** All components you add to or create in OracleAS Portal must belong to a Portal DB Provider. Refer to the *OracleAS Portal online help*, for more information on how to create a Portal DB Provider.

7. Click **Next**.

8. On the resulting page, fill in desired values:

   ■ In the **OS Printer Name** field, enter the operating system printer name, for example:

   UNIX: *printer_name*

   Windows: *printer_server**printer_name* (for a remote printer)
   printer_name (for a local printer)

   This printer *must* be available to the Reports Server.

   > **Note:** Printer availability is set via the operating system on the Report Server's host machine.

   ■ (Optional) Enter a description of the Printer in the **Description** field.

9. Click **Next**.

10. (Optional) Select an Availability calendar to restrict the days and times the printer can be used.

> **See Also:** Creating an Availability Calendar
>
> For more information on how to create an Availability Calendar.

11. Click **Finish**.

The resulting page summarizes your settings for this printer. On this page, you can edit your settings, get detailed registration information about the printer, or delete it altogether.

> **See Also:** The Manage Portlet
>
> For more information on the fields and descriptions listed in the Manage portlet, i.e., Develop, Manage, and Access tabs.

12. Click **Close** to close this page and return to OracleAS Portal's **Oracle Reports Security** page.

You have completed registering a printer with OracleAS Portal. This registration is meaningful for reports that are run through OracleAS Portal as well as those run outside of OracleAS Portal.

11.2.3 Registering a Reports Server

Before you can define access controls for the Reports Server, you *must* register your server within OracleAS Portal. Registration provides OracleAS Portal with the information it needs to identify and locate all available Reports Servers. This becomes particularly important when you register individual reports; during this process you are required to choose from a list of Reports Servers, and servers must be registered to appear on this list.

Table 11–2   Sample Values

Property	Sample Value
Name (internal name)	`myrep_server`
Display Name	`My Reports Server`
Portal DB Provider	`PORTAL_APP`
Reports Server Name	`rep_machine_name, e.g., rep_myserver1`
Oracle Reports Web Gateway URL for JSP reports	`http://myias.mycomp.com:7778/`

Table 11–2 Sample Values

Property	Sample Value
Oracle Reports Web Gateway URL for RDF reports	`http://myias.mycomp.com:7778/report s/rwservlet`
Availability Calendar	`COMCAL`

This section describes how to register Reports Servers in OracleAS Portal.

To register a Reports Server:

1. Log in as an administrator to OracleAS Portal.

2. Navigate to the Builder page.

3. Click the **Administer** tab.

4. Click the **Oracle Reports Security Settings** link in the **Oracle Reports Security** portlet. The **Oracle Reports Security** portlet enables you to use the security features in OracleAS Portal at the time of defining access to the server, printer, calendar, and reports definition file.

5. Click the **Create Reports Server Access** link in the Reports Server Access portlet.

6. On the resulting page, the **Name** (internal name) and the **Portal DB Provider** fields contain default values. To include custom values:

   ■ Enter a unique name in the **Name** field that will identify the Reports Server internally in OracleAS Portal, for example, MY_REPORTS_SERVER. This name must follow the OracleAS Portal rules for a valid component name, i.e.,:

       * It must be no more than 30 characters

       * It must contain only alphanumeric characters (no spaces or special characters allowed).

       * The first character must be a letter (not a number).

   ■ Enter the name you want to display for this server in the **Display Name** field. The Display Name is the name that is exposed to your users through OracleAS Portal.

> **Note:** The Display Name, unlike the internal Name, can have spaces in it.

■ Select the Portal DB Provider that will own the Reports Server from the Portal DB Provider list of values. The Portal DB Providers displayed are those in which you have privileges to build components.

> **Note:** All the components you add to or create in OracleAS Portal must belong to a Portal DB Provider. Refer to the *OracleAS Portal online help*, for more information on how to create a Portal DB Provider.

7. Click **Next**.

8. On the **Server Definition** page:

   ■ Enter the name of the Reports Server in the **Reports Server Name** field. This is the unique name assigned to the server at the time of installation, i.e.,`rwserver -install` *repservername* or `rwserver server=`*repservername*.

   ■ (Optional) Enter a description for the Reports Server in the **Description** field.

   ■ Enter the URL location of your JSP files in the **Oracle Reports Web Gateway URL for JSP reports** field. The URL should be in the following format:

   `http://`*your_web_server.domain:port*`/`

   For example:

   `http://myias.mycomp.com:7779/`

   ■ Enter the URL location of your Reports Servlet in the **Oracle Reports Web Gateway URL for RDF reports** field. The URL should be in the following format:

   `http://`*your_web_server.domain:port*`/`*virtual_path_to_rwservlet*`/rwservlet`

   See Also: Chapter 3, "Configuring OracleAS Reports Services"

   For more information on specifying the virtual path.

For example:

```
http://myias.mycomp.com:7778/reports/rwservlet
```

- (Optional) Select the **Run Only Registered Report Definition Files** check box. This ensures that only the report definition files registered with OracleAS Portal can be executed on this Reports Server.

  Leave this box unchecked if you want this Reports Server to accept any report definition file, including those not registered in OracleAS Portal, as long as the user who submits the report request has access privileges to this Reports Server.

- Select the printer(s) that you want to make available to this Reports Server from the **Printers** list. Use control-click (Windows) or click (UNIX) to select multiple printers.

9. Click **Next**.

10. (Optional) Enter a Custom Destination Type, if you have defined a custom destination type.

   > **See Also:** Chapter 7, "Configuring Destinations for OracleAS Reports Services"
   >
   > For more information on custom destination types.

11. Click **Next**.

12. (Optional) Enter the Availability Calendar name or click the list button to select the Availability Calendar that determines the days and times this Reports Server is and is not available to accept report requests.

   > **See Also:** Creating an Availability Calendar
   >
   > For more information on how to create an Availability Calendar.

13. Click **Finish**.

   The resulting page summarizes your settings for this Reports Server. On this page, you can edit your settings, get detailed registration information about the Reports Server, or delete it altogether.

See Also: The Manage Portlet

For more information on the fields and descriptions listed in the Manage portlet, i.e., Develop, Manage, and Access tabs.

14. Click **Close** to close this page and return to the **Oracle Reports Security** page.

You have registered a Reports Server. Now you can register a report.

11.2.4 Registering a Report

Registering a report is a required step that allows you to define who can run a report, when a report is available to run, which server(s) can be used to process report requests, how a report is delivered, and the printer(s) to which a report can be sent.

In addition to using registration to designate which users have access to a report, you can also specify, via a OracleAS Portal parameter form, how users are to interact with the report.

User parameters are created in Reports Builder at the time of designing the report. You can assign values to these parameters when you run the report in OracleAS Portal.

> **Note:**   You can use the parameter settings available through OracleAS Portal to duplicate or create a subset of the parameters defined in Reports Builder at design time. At runtime, the Reports Server disregards any parameters that you set in OracleAS Portal not defined in Reports Builder at design time.

Registering a report within OracleAS Portal creates an OracleAS Portal component that can be deployed as a portlet through Portal. We recommend that you register only one instance of a report file in OracleAS Portal. If you define multiple OracleAS Portal report objects for one report, all are given security checks at runtime. If any of them fail the security check, then all fail, and the job will not run.

> **Note:** Running reports from within OracleAS Portal requires the HTML iframe tag, which is not supported in Netscape 4.x. As a result, the following limitations apply when using Netscape 4.x:
>
> - A report portlet cannot display in place if you are using HTTPS. You need to click on the portlet title to see the report in a separate browser window.
>
> - A report portlet cannot be scheduled to run via the Customize link if you are using HTTPS.

Table 11–3   Sample Values

Property	Sample Values
Name (internal name)	`Employee_Report`
Display Name	`Employee Report`
Portal DB Provider	`PORTAL_APP`
Oracle Reports File Name	`employee_report.jsp`
Execute	`as JSP`
Name (Optional Parameters)	`userid`
Display Name (Optional Parameters)	`User Identification`

To register a report:

1. Log in as an administrator to OracleAS Portal.

2. Navigate to the Builder page.

3. Click the **Administer** tab.

4. Click **Oracle Reports Security Settings link i**n the **Oracle Reports Security** portlet.

5. Click the **Create Reports Definition File Access** in the **Reports Definition File Access** portlet.

6. On the resulting page, the **Name** (internal name) and the **Portal DB Provider** fields contain default values. To include custom values:

- Enter a unique name in the **Name** field that will identify the report internally in OracleAS Portal, for example, MY_REPORT. This name must follow the OracleAS Portal rules for a valid component name, i.e.,:

  * It must be no more than 30 characters

  * It must contain only alphanumeric characters (no spaces or special characters allowed).

  * The first character must be a letter (not a number).

- Enter the name that you want to display for this report in the **Display Name** field. The Display Name is the name that is exposed to your users through OracleAS Portal.

  > **Note:** The Display Name, unlike the internal Name, can have spaces in it.

- Select the Portal DB Provider that will own the Reports Server from the Portal DB Provider list of values. The Portal DB Providers displayed are those in which you have privileges to build components.

  > **Note:** All the components you add to or create in OracleAS Portal must belong to a Portal DB Provider. Refer to the *OracleAS Portal online help*, for more information on how to create a Portal DB Provider.

7. Click **Next**.

8. Enter or select information as follows:

   - Select the Reports Server(s) to be available to run this report from the **Reports Servers** list of values. Use control-click (Windows) or click (UNIX) to select multiple servers.

   - Enter the report file name, including its extension in the **Oracle Reports File Name** field.

     The report definition file can be an `.rdf`, `.jsp`, or `.xml` file. If the path to this file is included in your REPORTS_PATH environment variable, do not enter it here. If the path is not included in REPORTS_PATH, include it here along with the filename. Do this for all report definition files except those

you will run as stand-alone JSPs. For JSPs, you need to define the name as *virtual_path*/*reportname*.jsp.

See Also:

- Appendix B, "Reports-Related Environment Variables"

For more information on Oracle Reports-related environment variables.

- Chapter 3, "Configuring OracleAS Reports Services"

For more information on specifying the virtual path.

- (Optional) Enter a description for this report in the **Description** field.

- Select either **via servlet** or **as JSP i**n the **Execute** field. The selection you make here will affect the choices that are available on the next wizard page.

    * **via servlet:** If you plan to run the report through the Reports Servlet.

    * **as JSP:** If you will run a deployed JSP report.

9. Click **Next**.

10. Select the Destination settings on the Required Parameters page.These settings are only applicable if you run your report through the Reports Servlet. At runtime, anywhere you have indicated multiple selections using control-click, a list of values will be offered to your users from which they can set their own runtime information:

    - Select the destination types from among Cache, File, Mail, OraclePortal, OracleWireless, Printer, FTP, WebDAV, or custom destination types. **Types** specifies the destination types acceptable for this report. If the server you associate with this report supports custom destination types, which you indicated when you registered the Reports Server in OracleAS Portal, the types you indicated will display on this list.

    - **Formats** defines the acceptable output format(s) for this report. Choose among HTML, HTMLCSS, PDF, XML, RTF, Delimited, PostScript, and Character

    - **Printers** specifies the registered printer(s) to which this report can be sent. The printers that appear on this list are determined by those you chose when you set up access to the Reports Server(s) you are associating with this report. When users choose a Reports Server on the runtime parameter

form, only those printers that are associated with the selected Reports Server and that are accessible to those users are listed.

11. Select the Parameter Form Template and click **Preview Template** to see what the selected template looks like:

   ■ **Parameter Form Template** specifies the template that will define the look and feel of the Portal parameter form from which you will run the report. This value is used only when the report is exposed through the Portal. Choose a template from the list of values.

   > **Note:** For information about adding your own templates to this list, see the *OracleAS Portal online help*.

12. Click **Next**.

13. Define the limits for the report's existing parameters on the **Optional Parameters** page:

   ■ Enter the name or user parameter to restrict the values available to users in the Name field. For example, SALES_REGION or COPIES.

   ■ Enter the display name of the system or user parameter. This name will be used to identify the parameter on the runtime parameter form.

   ■ Enter the name of the list of values, or select the values from a predefined list of values. The list must already exist. For information on creating a list of values, see the *OracleAS Portal online help*.

   ■ Enter the lowest value that you wish to set for a range of values in the Low Value field.

   ■ Enter the highest value that you wish to set for a range of values in the High Value field.

   ■ Click **More Parameters** if you wish to add more rows for additional parameters and values.

14. Click **Next**.

15. (Optional) Enter the Availability Calendar name or click the list button to select an existing Availability Calendar.

   Use the availability calendar to limit the days and times this report can be run.

See Also: Creating an Availability Calendar

For more information on how to create an Availability Calendar.

16. Click **Next**.

17. (Optional) Enter a validation trigger to create a programmatic restriction.

   Use validation triggers 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.

   The function that you specify as a validation trigger must return a boolean value (TRUE or FALSE). If the function returns TRUE, the job is run. If the function returns FALSE, an error message is displayed and the job is not run.

18. Click **Finish** to close the wizard and complete report registration.

   The resulting page summarizes your registration information and provides the opportunity to perform additional actions on your report.

   See Also: Publishing a Report in OracleAS Portal

   For more information on how to run your report from OracleAS Portal.

   ■ Click **Customize** to view the report's runtime parameter form.

   Table 11–4 summarizes the options available on this page.

Table 11–4 Options on the runtime parameter form

Option	Description
Run Report	Click to run this report with the specified parameter values.
Save Parameters	Click to save the parameter value selections.
Server	Select the Oracle Reports Server that you want to receive this report request. Only the servers that you chose at the time of registering the Report are displayed in this list box.
Printer	Select the printer that you want to print your report output. Only the printers that you chose at the time of registering the report are displayed in this list box.
Destype	Select the destination type. Only the destination types that you chose at the time of registering the report are displayed in this list box.

Table 11–4   Options on the runtime parameter form

Option	Description
Desformat	Select the destination format. Only the destination format that you chose at the time of registering the report are displayed in this list box.
Desname	Enter the name of the output file when destype=FILE, or enter the e-mail addresses when the Destype is MAIL. Separate multiple addresses with commas. The destination name is required when you choose FILE or MAIL as the destype.
SSOCONN	Enter one or more SSO connection strings. Separate multiple strings with a comma (but no spaces). For more information about SSOCONN, refer to SSOCONN.
Visible to user	Check each parameter that you want to make available in the runtime parameter form when users run this report request. If the box in not checked, then the parameter is not displayed to users.
CGI/Servlet Command Key	Optionally, enter the key from the cgicmd.dat file that identifies the command line to run for this report.
Portlet Width	Use this field to control the width of the portlet. You can enter the value as a percentage of the page (e.g., 90%) or in pixels (e.g, 700).
	If no value is specified, OracleAS Reports Services uses its default value (640 pixels wide).
Portlet Height	Use this field to control the height of the portlet. You can enter the value as a percentage of the page (e.g., 50%) or in pixels (e.g, 400).
	If no value is specified, OracleAS Reports Services uses its default value (320 pixels high).
Additional User Parameters	Use this field to enter additional user parameters. For example, you can use this field to enter the path and name of the distribution XML file that defines how this report should be distributed.
	Use the same syntax you would use to specify these values in a command line request or within the cgicmd.dat file. If you wish to enter multiple additional parameters, simply separate each entry with a space.
	For more information about the distribution XML file, see Chapter 15, "Creating Advanced Distributions".

11.2.5 The Manage Portlet

Use the Manage portlet page to perform actions on existing Oracle Portal portlets; for example, executing, editing, copying, dropping, or viewing information about the portlet.

The actions you can perform on the portlet depend on your privileges. Also, not all actions listed here are available for all portlets. The name of the portlet on which you can perform these actions appears in the upper left corner of the page.

Table 11–5 details the fields and descriptions listed in the **Develop** tab.

Table 11–5   The Develop Tab

Field	Description
(portlet Type and Name)	Displays the portlet's type and name; for example:
	Form (table) my_formfor a form
	based on a table called my_form.
Provider	Displays the name of the provider in which the portlet was created.
Version(s) Status (Not applicable to all portlets)	Displays all the versions of the portlet and the current status of each version. Click a status to edit the portlet version.
	Note: If there are no hyperlinks, you do not have privileges to edit the portlet.
Last Changed	Displays the name of the user who created or last edited the portlet, and the date and time when the portlet was created or last edited.
Run Link (Not applicable to all portlets)	Displays the URL for the procedure or procedures that, when executed, display the portlet. You can copy and paste this URL into another Web page to create a link to the portlet.
	Note: A procedure that executes the portlet without parameters has the suffix .show. A procedure that executes the portlet with parameters has the suffix .show_parms.
PL/SQL source (Not applicable to all portlets)	The portlet builder wizards create a PL/SQL package to represent each portlet:
	Package Spec: Displays the portlet's PL/SQL specification.
	Package Body: Displays the portlet's PL/SQL body.

Table 11–5   The Develop Tab

Field	Description
Call Interface (Not applicable to all portlets)	Click Show to display the arguments that a portlet can accept that the end user can change at runtime. Also shown are examples of calling the portlet from a PL/SQL Stored Procedure and via a URL. When you run the package containing the portlet in PL/SQL or by calling it from a URL, you can edit the call interface to accept different arguments.
	Note: To view portlet source code, you must have Customize or Execute privileges on the portlet or the provider that owns it.
Edit Data Link (Not applicable to all portlets)	Click to connect to the URL containing the data, and to see and edit that data.
Edit	Click to edit the most recent version of the portlet. For example, you can reselect any table columns on which the portlet is based, change any fields or text that appear in the portlet, or choose a new look and feel.
Edit as New	Click to create and then edit a new version of this portlet. The existing portlet version does not change.
Edit Data (Not applicable to all portlets)	Click to see the spreadsheet and be able to edit the data within it.
Run	Click to run the current PRODUCTION version of the portlet.
	Note: If a valid package for the portlet doesn't exist, the portlet will not run.
Run As Portlet	Displays how the portlet will look as a portlet in a portal window (may look different than a full page display).
Customize	Click to display the customization form for the portlet. The customization form enables you to specify values that will be used to display the portlet.
	Note: If the current portlet is a form, Browse appears instead of Customize on this page.
Add to Favorites	Click to add the portlet to the Favorites list on your Oracle Portal Home page.
About	Displays stored attributes for the portlet.
Delete	Click to drop the portlet from the database.

Table 11–6 details the fields and descriptions listed in the **Manage** tab.

Table 11–6   The Manage Tab

Field	Description
Show/Hide SQL Query Info (Not applicable to all portlets)	Select to display or hide the SQL Query when running the portlet, for debugging purposes.
Show Locks on this portlet (Not applicable to all portlets)	Displays any locks currently active on the portlet (for example, if somebody else is editing it).
Export	Click to export the portlet from the database.
Copy	Click to copy the portlet from the database.
Rename	Click to rename the portlet (within the same provider).
Generate	Click to compile the PL/SQL package.
Monitor	Click to view a chart of all requests for the portlet and the users who made the request.

Table 11–7, Table 11–8, Table 11–9, Table 11–10, Table 11–11, Table 11–12, and Table 11–13 details the fields and descriptions listed in the **Access** tab.

Table 11–7   Portal Access

Field	Description
Publish as Portlet (Not applicable to all portlets)	Click to make the portlet available as a portlet.
	Note: To publish the portlet as a portlet, you must have the Publish Portlet privilege and you must make the provider that owns the portlet available via Expose as Provider on the Access provider page (Manage tab).

Table 11–8   Privilege Mode

Field	Description
Inherit Privileges from Provider	Select to allow the provider access privileges to override the portlet access privileges.
	Clear the check box and click Apply to allow the portlet access privileges to override the provider access privileges. In the Grant Access section, you can selectively grant or remove portlet access privileges for different users or groups (for example, Manage, Edit, View, Customize, or Execute).
	Note: To grant portlet access privileges to a user or group, you must have Manage access privileges on the portlet or provider that owns the portlet.

Table 11–9   Grant Access

Field	Description
Grantee	Enter the user or group to whom you want to grant the provider access privilege.
Execute	Choose the privilege you want to grant.
Add	Click to grant the provider access privilege.

Table 11–10   Change Access

Field	Description
Grantee	Displays the Oracle Portal user or group to whom the privilege is assigned. Click Error! Unknown switch argument.next to a grantee to delete all privileges.If you want to grant privileges to all Oracle Portal users, choose Public as the Grantee.
Type	Displays whether the grantee is an Oracle Portal user or group.
Privilege	Displays the privilege currently granted. To change a privilege, choose a new one and click Apply.

Table 11–11   Cell Privilege Mode

Field	Description
Inherit Privileges from portlet	Select to allow the portlet access privileges to override cell access privileges.
(Not applicable to all portlets)	Clear the checkbox and click Apply to allow cell access privileges to override the portlet access privileges. In the Alter Access section, you can selectively change cell access privileges for different users or groups (for example, Manage, Edit, View, Customize, or Execute). **Note:** To alter cell access privileges for a grantee, you must have Manage access privileges on the portlet or provider that owns the portlet.

Table 11–12   Alter Access

Field	Description
Grantee (Not applicable to all portlets)	Enter the user or group to whom you want to grant the cell access privilege.
Alter (Not applicable to all portlets)	Click to alter cell access privileges.

Table 11–13   Cache Invalidation

Field	Description
Clear Cache	Clears the cached version of the data, so that the next data request will be filled from the database.

12

Clustering Reports Servers

A cluster is a virtual grouping of servers into a community for the purpose of sharing request processing efficiently across members of the cluster. Clustering in OracleAS Reports Services is peer-level, which means that all members of the cluster take equal responsibility for sharing and processing incoming requests. If one member is shut down, the other members carry on managing the request load. If the output is present in one member's cache, another member can use it. There is no single-point-of-failure, where one machine's malfunction brings the whole system down.

This chapter contains information about enrolling a server in a cluster and benefits of clustering servers together. It contains the following sections:

- Cluster Overview
- Setting Up a Cluster

12.1 Cluster Overview

Assume you have the following servers:

```
serverA.cluster1
serverB
serverC.cluster1
```

ServerA.cluster1 and serverC.cluster1 are members of the same cluster called cluster1. They cooperate to process requests from a client. If a client sends a synchronous request to serverA.cluster1 and it does not have an idle engine of the specific job type, then it checks to see if serverC.cluster1 does. If serverC.cluster1 does have an idle engine, then serverA.cluster1 passes the request to serverC.cluster1 for processing.

In this example, ServerB is a stand-alone server and cannot receive processing requests from other servers, nor can it send processing requests to other servers.

You can have an unlimited number of servers in a cluster. If a cluster member is shut down, then it redistributes its pending synchronous jobs to another server in the cluster. As long as one server in the cluster is running, the cluster is working.

When the cluster is making its decision as to where an upcoming scheduled or immediate request should be processed, it prioritizes according to the following criteria:

1. Does any server in the cluster have information in cache that matches the request?

2. Is there a current, similar job in the queue?

3. Is an idle engine of the particular job type available?

4. Is the number of currently active engines less then the MAXENGINE number specified for the server for that job type?

Both stand-alone and clustered servers share the same, basic configuration. The cluster has no special configuration requirements, beyond needing to share a common cluster name and common public and private keys. There are no limitations on the platform used, the number of servers in the cluster, or the location of the server. There is no requirement to share resources within the cluster servers.

Engine output is locally cached in a particular Reports Server within the cluster, but it is also known and available to the entire cluster. If a server is down, that server's cached files are no longer available for reuse. This means that another server within the cluster must rerun the request to obtain the output. When the server is running again, all of the cached files become available due to the persistent state of the cache.

12.2 Setting Up a Cluster

Clustering in OracleAS Reports Services is as easy as naming all member servers with the same "dot extension," for example *server_name*.cluster or *server_name.xyz*, and ensuring that all member clusters share the same public and private key.

This section covers renaming your Reports Server, creating and specifying public and private keys, and submitting requests to a cluster. It contains the following sections:

- Renaming a Reports Server

- Generating New Public and Private Keys
- Entering Public and Private Keys in the Server Configuration File
- Restarting the Reports Server
- Submitting a Request to a Cluster

12.2.1 Renaming a Reports Server

It is likely that you are reading this material after you've already set up at least one Reports Server. If this is the case, you'll need to change the name of your server to add the cluster name to the server name.

> **Note:** If you haven't yet installed your servers, when you do install them you must give them all different server names but the same cluster name, for example servernameA.cluster1, servernameB.cluster1.

To rename a Reports Server:

1. If the server is running, shut it down:

   - If it's running on Windows as a service, stop it through the Services control panel.

   - If it's running on Windows through a server executable, or on UNIX through a shell script, click the **Shutdown** button in the Oracle Reports Server dialog box.

   - If it's running from a command line on Windows or UNIX, at the command prompt enter the following command for Windows or UNIX:

   This shuts down the server normally:

   ```
   rwserver server=server shutdown=normal authid=admin/pword
   ```

   This shuts down the server immediately:

   ```
   rwserver server=server shutdown=immediate authid=admin/pword
   ```

   This shuts down the server without displaying any related messages:

   ```
   rwserver server=server shutdown=normal authid=admin/pwrd batch=yes
   ```

The keywords used with the rwserver command are described in
Appendix A, "Command Line Options".

2. If you have custom configuration settings in your Reports Server configuration
   file (*server_name*.conf), rename this file to the new cluster name (*server_
   name.cluster_name*.conf).

   You'll find the configuration file in the following path on UNIX and Windows:

   ORACLE_HOME\reports\conf*server_name.conf*

   If you don't have custom configuration settings in your Reports Server
   configuration file, a new configuration file with the new name will be generated
   automatically when you restart the renamed server(s).

3. Rename the Reports Server in all affected files, giving each cluster member the
   same cluster name.

   - Open the servlet configuration file (rwservlet.properties) and
     respecify the server name to include the name of your cluster. For example:

     SERVER=*server_name.cluster_name*

     You'll find the servlet configuration file on both Windows and UNIX in the
     same path:

     ORACLE_HOME\reports\conf\rwservlet.properties

   - If you run the server as a Windows service, to rename the server you must
     uninstall and reinstall the service:

     To uninstall the Windows service, at the command prompt enter:

     rwserver -uninstall *server_name*

     To reinstall the Windows service, at the command prompt enter:

     rwserver -install *server_name.cluster_name*

   Note: Reinstalling the server also starts it up. You may want to
   shut it down until you have renamed all server cluster members,
   then start them all up together once you've set up your cluster.
   You'll find information on shutting the server down in Chapter 2,
   "Starting and Stopping OracleAS Reports Services".

Before you restart your Reports Server(s), you may generate server public and private keys and enter the resulting information in each member server's configuration file. How to do this is discussed in the next sections.

12.2.2 Generating New Public and Private Keys

The server public and private key files aid with message encryption and authentication between cluster members. The default files are stored in the `rwrun.jar` file in the following path (on both UNIX and Windows):

`ORACLE_HOME\reports\jlib\rwrun.jar`

Each member of a cluster must have the same public and private key files specified in their configuration files (`server_name.cluster_name.conf`). To ensure that your cluster members share exclusive public and private key files, generate new versions of them when you set up your cluster. Servers that will not be members of the cluster can go on using the default keys provided with OracleAS Reports Services.

To generate new public and private key files, at the command prompt, enter the following command:

`rwgenkey.sh public_key_file_name private_key_file_name (UNIX)`

`rwgenkey.bat public_key_file_name private_key_file_name (Windows)`

You can generate these files to specific directories by specifying the desired path in the command line along with the new public and private key file names. If you just specify the file name in the command line, the key files will be generated in the current directory.

12.2.3 Entering Public and Private Keys in the Server Configuration File

Once you generate new public and private key files, you must enter that information into all cluster members' Reports Server configuration files. You'll find each cluster member's version of this file in the following path for both UNIX and Windows on each server's host machine:

`ORACLE_HOME\reports\conf\server_name.cluster_name.conf`

To change public and private key files, go to the `connection` element in the server configuration file, and change (or add) entries for the `cluster` sub-element as follows:

```
<cluster publicKeyFile="path and filename of new public key"
privateKeyFile="path and filename of new private key">
```

You'll find more information about the `connection` element in Chapter 3, "Configuring OracleAS Reports Services".

12.2.4 Restarting the Reports Server

Once you have renamed your cluster members and respecified a common public and private key for each, you may start up your Reports Servers to activate the cluster.

To start up a Reports Server:

- If you're starting the Reports Server as a Windows service, open the Service control panel, and start the service.

- If you're starting the Reports Server from a command line, at the command prompt, enter the following command:

    On **Windows**:

    ```
    rwserver server=server_name.cluster_name
    ```

    On **UNIX**:

    ```
    rwserver.sh server=server_name.cluster_name
    ```

Once you've renamed your cluster members, re specified your public and private keys, and restarted your Reports Servers, you've completed the process of setting up your cluster.

12.2.5 Submitting a Request to a Cluster

To submit a request to a cluster:

In the Reports Servlet or JSP, specify:

```
server=cluster_name
```

For example, if you have two cluster members—one named mercury.cluster1, the other named venus.cluster1—then your `server` entry would be:

```
server=cluster1
```

The Reports Servlet or JSP will find a running Reports Server in the cluster and send the request to that Reports Server. Depending on the cache match or the server load, that Reports Server will either handle the request or redirect it to another server in the cluster.

Part II

Sending Requests to the Server

Part II provides detailed, practical information about publishing reports, including how to run requests; how to set up sophisticated, automatic report distributions; how to customize reports at runtime via XML customization files, and how to use database triggers to automatically invoke reports.

Part II includes the following chapters:

- Chapter 13, "Running Report Requests"
- Chapter 14, "Using the Oracle Reports Web Service"
- Chapter 15, "Creating Advanced Distributions"
- Chapter 16, "Customizing Reports with XML"
- Chapter 17, "Using Event-Driven Publishing"

13

Running Report Requests

This chapter discusses various ways to send report requests to the Reports Server. It includes the following sections:

- The Reports URL Syntax
- Report Request Methods
- Deploying Your Reports
- Publishing a Report in OracleAS Portal
- Specifying a Report Request from a Web Browser
- Sending a Request to the URL Engine
- Running Reports through a Web Service
- Scheduling Reports to Run Automatically
- Additional Parameters
- Reusing Report Output from Cache
- Using a Key Map File

13.1 The Reports URL Syntax

This section provides quick reference information on formulating a URL for publishing a report. It covers three deployment types:

- Servlet
- JSP
- CGI (for backward compatibility only)

The information is largely the same for both Windows and UNIX environments. Differences are noted.

13.1.1 Servlet

The syntax for the URL of a report run via the Reports Servlet is:

```
http://web_server.domain_name:port/alias/rwservlet?parameters
```

Table 13–1 lists and describes the components of the servlet URL.

Table 13–1   Components of a URL that calls the Reports Servlet

Component	Description
web_server	The name you gave the Oracle HTTP Server when you installed it.
domain_name	Your organization's domain name.
port	The port number on which the Oracle HTTP Server listens for requests. When no port is specified, the default is used (80).
alias	The virtual path that stands in for the absolute path to the files a URL will access.
rwservlet	Invokes the Reports Servlet.
?	Identifies the beginning of the command line options.
parameters	All the command line options, or the key to the key map file where command line options are specified.

The URL that calls the Reports Servlet could look like this:

```
http://neptune.world.com:80/reports/rwservlet?keyname
```

Keyname refers to a command line listed under a unique header (the key name) in the cgicmd.dat file. Note that this works differently for JSP files, which use the keyword/value pair cmdkey=value to specify key names for command lines that are stored in the cgicmd.dat file. You'll find more information about using key mapping in Using a Key Map File.

Using the servlet does not mean that you cannot also use JSP report files, if the JSP files contain both Web and paper layouts. When you run the report, specify the servlet in the URL and call the JSP with the command line option: report=myreport.jsp.

For example:

```
http://neptune.world.com:80/reports/rwservlet?report=myreport.jsp&destype=cache&
desformat=html
```

You'll find more information about command lines in Appendix A, "Command Line Options".

Note: Yo can also supply these parameters within the JSP file itself.

13.1.2 JSP

The syntax for a JSP-based report URL is:

`http://`*web_server.domain_name:port/alias*`/myreport.jsp?`*parameters*

Table 13–2 lists and describes the components of the JSP-based report URL.

Table 13–2   Components of a JSP-based Report URL

Component	Description
web_server	The name you gave the Oracle HTTP Server when you installed it.
domain_name	Your organization's domain name.
port	The port number on which the Oracle HTTP Server listens for requests. When no port is specified, the default is used (80).
alias	The virtual path that stands in for the absolute path to the files a URL will access.
myreport.jsp	The report *.jsp file[s] that you want this URL to execute.
?	Identifies the beginning of the command line options.
parameters	All the command line options, and/or the key to the key map file where command line options are specified.

The URL used to invoke a JSP-based report could look like this:

`http://neptune.world.com:80/jsp/myreport.jsp?`

You can specify a key in the URL that refers to a command line in the cgicmd.dat file that contains additional command line parameters. In this case, you must use the name value pair: cmdkey=*keyname*. This can appear anywhere in your URL,

provided it follows the start of the query string (marked by a question mark). For example:

```
http://neptune.world.com:80/jsp/myreport.jsp?userid=scott/tiger@hrdb&cmdkey=key1
```

In your URL, use an ampersand (&) with no spaces to string parameters together.

Using a JSP does not mean that you cannot also use the Reports Servlet. When you run the report, specify the servlet in the URL and call the JSP with the command line option: report=*myreport*.jsp.

For example:

```
http://neptune.world.com:80/reports/rwservlet?report=myreport.jsp&destype=cache&
desformat=html
```

You'll find more information about command line keywords in Appendix A, "Command Line Options". You'll find more information about the cgicmd.dat file in Using a Key Map File.

13.1.3 CGI

> **Note:** The Reports CGI is included in Oracle Application Server Reports Services for backward compatibility. We strongly recommend that you deploy your reports with either a servlet or JSP implementation.

The syntax for the URL of a report run via the Reports CGI on **Windows** is:

```
http://web_server.domain_name:port/alias/rwcgi.exe?parameters
```

And on **UNIX** is:

```
http://web_server.domain_name:port/alias/rwcgi.sh?parameters
```

Table 13–3 lists and describes the components of a CGI-based report URL.

Table 13–3   Components of a URL that Calls the Reports CGI

Component	Description
web_server	The name you gave the Oracle HTTP Server when you installed it.
domain_name	Your organization's domain name.

Table 13–3   Components of a URL that Calls the Reports CGI

Component	Description
port	The port number on which the Oracle HTTP Server listens for requests. When no port is specified, the default is used (80).
alias	The virtual path that stands in for the absolute path to the files a URL will access.
rwcgi.exe	The executable file that invokes the CGI component of OracleAS Reports Services. If OracleAS Reports Services is installed on a UNIX machine, use ".sh" in lieu of ".exe".
?	Identifies the beginning of the command line options.
parameters	All the command line options, or the key to the key map file where command line options are specified.

The URL used to invoke a CGI implementation could look like this on **Windows**:

```
http://neptune.world.com:80/cgi-bin/rwcgi.exe?key2
```

And like this on **UNIX**:

```
http://neptune.world.com:80/cgi-bin/rwcgi.sh?key2
```

13.2 Report Request Methods

There are a number of request methods available to you for running your report requests. These include:

- **The** rwclient **command line**

  The rwclient command line (rwclient.sh on UNIX) is available for running report requests from a command line in a non-Web architecture. It references an executable file that parses and transfers the command line to the specified Reports Server. It can use command line options similar to those used with the Reports Runtime executable file, rwrun (rwrun.sh on UNIX).

  On Windows, a typical rwclient command line request looks like this:

  ```
  rwclient report=my_report.rdf userid=username/password@my_db server=server_
  name destype=cache desformat=html
  ```

  On UNIX, the same command would look like this:

  ```
  rwclient.sh report=my_report.rdf userid=username/password@my_db
  server=server_name destype=cache desformat=html
  ```

See Command Line Options for more information about command line options.

- **A URL**

  To run a report from a browser, use the URL syntax. The Reports Servlet (and CGI, for backward compatibility) converts the URL syntax into an `rwclient` command line request that is processed by Oracle Application Server Reports Services. You can give your 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.

- **Via OracleAS Portal**

  The OracleAS Portal component enables you to add a link to a report in an OracleAS Portal page or portlet, or to output report results directly into a portlet. Each report link points to a packaged procedure that contains information about the report request. OracleAS Reports Services system administrators use OracleAS Portal wizards to create the packaged procedure making it more convenient and secure to publish the report via the Web. Authorized users accessing the OracleAS Portal page group simply click the link to run the report. System administrators can run the report directly from the wizard. See the *OracleAS Portal online help* for more information.

  Refer to Publishing a Report in OracleAS Portal for more information about how to publish your report as a portlet.

- **A packaged procedure**

  `SRW.RUN_REPORT` is a built-in that runs a Reports Runtime command. When you specify `SRW.RUN_REPORT`, set the SERVER option to the Reports Server name to cause the `SRW.RUN_REPORT` command to behave as though you executed an `rwclient` command.

  Refer to the *Reports Builder online help* for more information.

- **A Web service**

  You can expose OracleAS Reports Services as a Web service and then call it from any Web service aware environment (e.g., a Java application).

13.3 Deploying Your Reports

Once you've created your report, you can deploy it so that end users can view it. This section describes how to deploy a report with a paper layout (i.e., REP, RDF,

XML, or JSP report) and how to deploy a report with a Web layout (i.e., a JSP report).

> **Note:** For an example on building and testing a JSP-based Web report, refer to the *Oracle Reports Tutorial* and the "Building a JSP-Parameter Form for a Web Report" chapter in the *Oracle Reports Building Reports* manual.

The following table describes which method you can use to deploy your report, depending on the type of report.

Table 13–4   Methods for Deploying a Report

Type of Report	Method	Reason for Using
Report with paper layout (REP, RDF, XML)	Deploying a Report with a Paper Layout	Method for deploying a report with only a paper layout.
JSP report with a paper layout	Deploying a Report with a Paper Layout	Simplest method for deploying a paper report of any type. However, if the JSP report has both a paper and Web layout, we recommend you refer to Deploying a JSP report to the Web and to Paper.
JSP report with a paper and Web layout	Deploying a JSP report to the Web and to Paper	Strongly recommended for those who want to publish a report to both the Web and to paper.

13.3.1 Deploying a Report with a Paper Layout

Once you've created your paper report, you can deploy it to the Reports Server so that users can run the report. The steps in this section show you how to deploy a report of type RDF, REP, XML or JSP.

> **Note:** JSP reports can be deployed either to the Web or to paper, depending on the layout the report designer used for the JSP report. This section discusses how to deploy a JSP report with a paper layout. If you want to deploy a JSP report with a paper and Web layout, follow the steps in Deploying a JSP report to the Web and to Paper.

To deploy your paper report:

1. Transfer the report file (RDF, REP, XML, or JSP) and its associated files (e.g., PLL, PLX or referenced images) to the deployment directory on your application server.

Note: To transfer the file, you can use any method available, such as FTP or WebDAV.

Make sure the directory on the application server where you've transferred the file is listed in the Reports Server access path. If it is not, use the REPORTS_PATH environment variable, or set the sourceDir property of the Reports engine element in the server configuration file.

13.3.2 Running a Report with a Paper Layout

Now that you have deployed your paper report, you can run it from a Web browser.

In a browser, for example, you can type the following URL in the Location field:

```
http://your_web_server:port_num/rwservlet?server=server_
name&report=myreport.rdf&userid=username/password@my_db&desformat=pdf
```

Your report displays as a PDF (since in this case desformat=PDF) in the browser.

For more information on running a report from the browser, refer to Specifying a Report Request from a Web Browser.

13.3.3 Deploying a JSP report to the Web and to Paper

There are two ways you can deploy your JSP reports: through the existing Oracle Reports application, or through a J2EE application you create yourself. Using an existing application is useful when you are developing and testing your JSP-based Web reports. When you are ready to deploy your reports, however, we recommend you use an application you've created yourself.

About JSP reports with both paper and Web layouts

With Reports Builder, you can create a JSP report with a paper layout, a Web layout, or both. You execute these reports using different processes:

- JSP reports with paper layouts are executed via the Reports Engine
- JSP reports with Web layouts are executed via the J2EE container

If your report depends on Java classes (e.g., Barcode classes, a Web Service stub, etc.), you must configure the process to access these classes. That is, if your JSP report with a paper layout contains a Java class, you must set the classPath property of the Reports engine element in the server configuration file (*ORACLE_HOME*\reports\conf*server_name*.conf).

If your JSP report with a Web layout contains a Java class, you can either add the classes or JAR to the WAR file, or change the J2EE container classpath. For more information, refer to the *Oracle Application Server Containers for J2EE* documentation.

> **Note:** For an example on building a report with a paper and Web layout, refer to the "Building a Report with a Barcode" chapter in the *Oracle Reports Building Reports* manual. For a simple JSP-based Web report example, refer to the *Oracle Reports Tutorial*.

The steps in this section show you how to deploy a JSP report with a paper and Web layout using a J2EE application. To deploy your JSP report with a paper and Web layout, you can create a new Oracle Reports J2EE application in your Oracle Application Server. You can create this application in a an existing instance or a new instance of Oracle Application Server Containers for J2EE (OC4J).

13.3.3.1 Creating a New J2EE Application

In this section, you will create a new J2EE application for Oracle Reports. You will create a Web application archive (a WAR file) that will contain the application information, then deploy it as an Enterprise archive (an EAR file). To create a new J2EE application, you can use Oracle JDeveloper, another Java development tool, or you can create it manually. If you do not use Oracle JDeveloper to create the application, you will need to make a few modifications to the application, as well as to your JSP report.

To create a J2EE application:

> **Note:** If you are not familiar with creating a J2EE application, refer to Sun's Web site at,(http://java.sun.com/j2ee/). For more information on using Oracle JDeveloper, refer to the *Oracle JDeveloper Online Help*.

1. Before you create your WAR file, make sure your application contains all the necessary directories, such as WEB-INF and the web.xml file.

   > **Note:** The WEB-INF directory must contain the JSP tag library for Oracle Reports, called reports_tld.jar. In Oracle Developer Suite, you can find the tag library here:
   >
   > *ORACLE_HOME*\reports\j2ee\reports_ids\web\WEB-INF\lib
   >
   > where *ORACLE_HOME* is the directory where the Oracle Developer Suite is installed.
   >
   > In Oracle Application Server, you can find the tag library here:
   >
   > *ORACLE_HOME*\j2ee\OC4J_BI_
   > Forms\applications\reports\web\WEB_INF\lib.

2. Make sure your JSP-based Web report points to the location of the JSP tag library for Oracle Reports. Otherwise, the report will not run.

3. Create a new WAR file, either manually or using a tool, such as Oracle JDeveloper. Make sure you create the WAR file according to the appropriate J2EE format.

4. If your JSP report contains a paper layout and you want to deploy your report to paper, open the web.xml file.

   > **Note:** In Oracle Developer Suite, the web.xml file is located here:
   >
   > *ORACLE_HOME*/reports/j2ee/reports_ids/web/WEB-INF
   >
   > On Oracle Application Server, the web.xml file is located here:
   >
   > *ORACLE_HOME*/j2ee/OC4J_BI_
   > Forms/applications/reports/web/WEB-INF .

   If you are deploying a JSP report that only contains a Web layout, continue to Step 7.

5. Add the following code to the web.xml file.

   ```
   <servlet>
   <servlet-name>rwservlet</servlet-name>
   ```

```
      <servlet-class>oracle.reports.rwclient.RWClient</servlet-class>
      <load-on-startup>yes</load-on-startup>
</servlet>
<servlet-mapping>
      <servlet-name>rwservlet</servlet-name>
      <url-pattern>/rwservlet*</url-pattern>
</servlet-mapping>
```

This new definition will redirect all URLs ending with /rwservlet to the servlet you've defined.

> **Note:** You can change the servlet name and URL.

6. Save the web.xml file.

7. Create an EAR file from the WAR file. Once these files are compiled, note where they are saved.

13.3.3.2 Deploying the Application Using OC4J

After you've created the WAR and EAR files, you can deploy them to the Oracle Application Server, which will serve the application to the Web. You can deploy these files using Oracle Enterprise Manager using either an existing OC4J instance or a new OC4J instance.

This section contains the two methods of deploying the J2EE application:

- Deploying the J2EE Application Using an Existing OC4J Instance
- Deploying the J2EE Application in a New OC4J Instance

13.3.3.2.1 Deploying the J2EE Application Using an Existing OC4J Instance

1. Make sure you've created the J2EE application as described in Creating a New J2EE Application.

2. In Oracle Enterprise Manager, display the detail page for your middle tier.

3. Under System Components, click **OC4J_BI_Forms**.

4. Under Deployed Applications, click **Deploy EAR file** to deploy the EAR file you created in Creating a New J2EE Application.

5. On the first page of the Deploy Application wizard, click **Next** (located at the bottom of the screen).

6. On the Select Application page, under **Select the J2EE application (EAR file) to be deployed**, enter the location of the EAR file you created in the previous section.

7. Under **Specify a unique application name for this application**, type the name of your application, such as `MyReportApp`, then click **Next**.

8. On the URL Mapping page, note that the text in the **URL Binding** field is the name your users will enter to access the new application.

9. In the URL Binding field, add a forward slash (/) to the beginning of the application name, since it is part of a URL address. For example:

   `/MyReportApp`

10. Click **Finish**.

11. On the next page, click **Deploy**.

12. On the OC4J_BI_Forms detail page that displays, you should now see your application (`MyReportApp`) listed under **Deployed Applications**.

13. Click your application name (`MyReportApp`).

14. On the Application page, under Properties, click **General**.

15. Under Library Paths, click **Add Another Row**, then add the following path to the `rwrun.jar` library:

   ORACLE_HOME\reports\jlib\rwrun.jar

16. Add another row with the following path to the `zrclient.jar` library:

   ORACLE_HOME\jlib\zrclient.jar

17. Click **Apply**, then click **OK**.

18. Click **Stop**, then **Start** to restart your application so that the new library paths take effect.

13.3.3.2.2 Deploying the J2EE Application in a New OC4J Instance

1. Make sure you've created the J2EE application as described in Creating a New J2EE Application.

2. In Oracle Enterprise Manager, display the detail page for your middle tier.

3. Click **Create OC4J Instance**.

4. Type the name of your OC4J instance.

5. Click **Create**.

6. On the confirmation page, click **OK**.

7. On Application Server page, under System Components, you should now see the new OC4J instance.

   Now, you must manually configure the OC4J to support connection to a Reports Server and the security integration.

8. Copy the following properties and their definitions in the `oc4j.properties` file from an existing OC4J instance, for example the OC4J_BI_FORMS instance (`ORACLE_HOME`/j2ee/OC4J_BI_FORM/config/oc4j.properties), into the `oc4j.properties` file of your new OC4J instance (`ORACLE_HOME`/j2ee/`your application`/config/oc4j.properties):

   - `oracle.home`

   - `java.rmi.server.randomIDs`

   - `oracle.display`

   - `oracle.path`

   - `org.omg.CORBA.ORBClass`

   - `org.omg.CORBA.ORBSingletonClass`

9. In the `opmn.xml` file in your `ORACLE_HOME`, add the `PATH` and `DISPLAY` properties to your new OC4J instance:

   a. In `ORACLE_HOME`/opmn/conf/opmn.xml, find the XML element that describes your new OC4J instance:

   ```
   <oc4j instanceName="<your application>" gid="<your application>">
         <config-file path="D:\oracle\Ora9iasR2App\j2ee\<your
   application>\config\server.xml"/>
         <oc4j-option value="-properties"/>
         <port ajp="3001-3100" rmi="3101-3200" jms="3201-3300"/>
      </oc4j>
   ```

   b. Add the `PATH` and `DISPLAY` properties by copying them from the OC4J_BI_FORMS instance in the same `opmn.xml` file, for example:

   ```
      <oc4j instanceName="<your application>" gid="<your application>">
         <config-file path="D:\oracle\Ora9iasR2App\j2ee\<your
   application>\config\server.xml"/>
         <oc4j-option value="-properties"/>
         <port ajp="3001-3100" rmi="3101-3200" jms="3201-3300"/>
   ```

```
<environment> <!-- entry to copy from the other instan -->
  <prop name="PATH" value="values are dependant of the system"/>
  <prop name="DISPLAY" value="localhost:0"/>
</environment>
</oc4j>
```

c. Restart the OC4J instance

13.3.4 Running a JSP-Based Web Report from a Browser

If your JSP report is a Web report, you can now run your JSP-based Web report from a Web browser. In a browser, type the following URL in the Location field:

```
http://your_computer_name:port/MyReportApp/JSPreportname.jsp?userid=user
ID/password@database_name
```

> **Note:** In the above URL, *MyReportApp* is the name of the application you created.

If you wish you modify your JSP-based Web report at this point, you can either:

- Replace the report in the above location.
- Recreate the WAR file with the modified JSP-based Web report, then redeploy the application. For more information, refer to Creating a New J2EE Application.

For more information on running a report from a browser, refer to Specifying a Report Request from a Web Browser.

13.3.5 Running a JSP report with a Paper Layout

If your JSP report has a paper layout, you can now run your JSP report from a browser using the following URL:

```
http://your_web_server:portnum/MyReportApp/rwservlet?report=
myreport.jsp&userid=username/password@my_db&server=server_
name&desformat=pdf&destype=cache
```

Your report displays as a PDF (since in this case desformat=PDF) in the browser.

For more information on running a report from a browser, refer to Specifying a Report Request from a Web Browser.

13.3.6 Running with the WE8MSWIN1252 character set on Solaris

There are no UNIX fonts built into the `WE8MSWIN1252` character set. This may cause Oracle Reports to fail when `NLS_LANG=AMERICAN_ AMERICA.WE8MSWIN1252.` Therefore, you must map the code page of the installed fonts (defined in the `Tk2Motif.rgb` file) to the `WE8MSWIN1252` character set. `TK2Motif.rgb` is located in the *ORACLE_HOME*/guicommon9/tk90/admin/ directory.

> **Note:** This mapping is required for Reports Builder, Reports Converter in non-batch mode (`batch=no`), Reports Server / Reports Runtime with `REPORTS_DEFAULT_DISPLAY=NO`. Reports Server / Reports Runtime uses `REPORTS_DEFAULT_DISPLAY` to determine the fonts needed.

Example1:

`Tk2Motif*fontMapCs: ISO8859-1=WE8MSWIN1252` (if there are ISO8859-1 fonts installed on the system.)

13.4 Publishing a Report in OracleAS Portal

One of the best ways to publish your report is through the declarative, secure interface of OracleAS Portal.

> **See Also:**
>
> Registering a Reports Server
>
> Registering a Report
>
> Registering a Printer
>
> Creating an Availability Calendar
>
> For more information on registering required components with OracleAS Portal before proceeding.

> **Note:** When you use features like OracleAS Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by OracleAS Reports Services may override the initial NLS_LANG setting. This change may in turn affect the behavior of the running report, such as bidirectional output in PDF. On UNIX platforms, you can work around this issue by setting the NLS_LANG explicitly in report.sh. You can also use the new environment switching functionality to dynamically set the environment for reports. Refer to Dynamic Environment Switching, for more information.

To a expose a report in a portal, you must do the following:

1. Create a provider for your reports. This step defines a provider to contain the reports you wish to make available to users in the portal.

2. Create the report definition file access**. This step makes the report available as a portlet to page designers within the portal by defining the reports properties, in particular the provider that contains it.

3. Add the report as an item link[1] or as a portlet[2] to a page and optionally customize it. This step makes the report available to users on a page and enables the page designer to set the report parameters and schedule it to run automatically.

13.4.1 Creating a Provider for Your Reports

If you do not already have a provider defined to contain your reports, you need to create one. For more information on creating a provider, see the *OracleAS Portal online help*.

> **Note:** The provider that contains your reports must be a database provider and must have the Expose as Provider setting selected on its Access page.

---

[1] An individual piece of content (text, hyperlink, image, etc.) that resides on a page in an item region.

[2] A reusable, pluggable Web component that typically displays portions of Web content.

13.4.2 Creating the Report Definition File Access

To make your report available as a portlet, you must do the following:

> **Note:** If you need to create report definition file access for a number of reports, it may be more efficient to batch register them. For more information, see Batch Registering Reports in OracleAS Portal.

1. If you are not already on the Builder page, click **Builder** at the top of the page.

2. Click the **Administer** tab.

3. In the Oracle Reports Security portlet, click **Oracle Reports Security Settings**.

4. In the Reports Definition File Access portlet, click **Create Reports Definition File Access**.

5. Follow the steps in the wizard and click the question mark in the upper right corner for additional information about the available settings. At the end of the wizard, click **Finish**.

6. Click the **Access** tab.

7. Click **Publish to Portal**.

8. Click **Apply**. Your report has now been added to the Portlet Repository and you can add it to a page.

13.4.3 Adding the Report Portlet to a Page

Once the portlet for your report is in the Portlet Repository, you may add it to any page just as you would any other portlet.

1. If you are not already on the Builder page, click **Builder** at the top of the page.

2. Click the **Build** tab.

3. In the Page Groups portlet, choose the name of the page group in which you want to place your report portlet.

4. Create a new page by clicking **Create a Page** or edit an existing page by entering the name of an existing page and clicking **Edit**.

5. If you are creating a new page, follow the steps in the wizard and click the question mark in the upper right corner for additional information about the available settings. Click **Finish** when you are done.

> **See Also:** Adding the Reports Component as an Item to a Page
> for information on how to add the Oracle Reports item to a page.

6. If you are editing an existing page, skip to the next step.

7. In the page region where you wish to add your report portlet, click the **Add Portlet** tool.

> **Tip:** Hints for each tool will display when you roll your mouse over them.

8. Drill down through the Portlet Repository to the provider that contains the report portlet. The report portlet is listed in the Portlet Repository under the Portal DB Provider to which it belongs. The location of the provider depends on how the Portlet Repository has been organized. If the Portal DB Provider is a fairly new provider, it may be under the New page of the Portlet Repository.

9. Click the name of your report portlet to add it to the **Selected Portlets** list.

10. Click **OK**.

11. Click **Customize** in the upper right corner of your report portlet.

12. Enter parameter values in the **Parameter** tab and, if desired, schedule the job to run automatically in the **Schedule** tab.

13. You can control the size of the portlet by specifying the **Portlet Width** and **Portlet Height** parameters on the Customize page for the Reports Definition File object. The value of these parameters may be a percentage (%) or a number of pixels.

For example, you can enter:

Portlet Width: 90%

Portlet Height: 480

If no value is specified, Oracle Application Server Reports Services uses its default value (640 pixels wide and 320 pixels high).

If the **Portlet Width** and **Portlet Height** fields are visible to users, then they can also adjust each portlet's width and height via Customize. The user's value will

override the value set in the Customize page of the Reports Definition File Object component.

14. You can choose whether to make a report's parameters visible to users through the Customization page of a Reports Definition File Access component.

To make a report's parameters visible to users:

a. Click **Customize** at the bottom of the Manage Component page for the report.

b. Click **Visible to user** for each parameter you want to expose.

> **Note:** You can also set the default value of the parameter from this page.

The value from the Manage Component page will always be used as the default value in the Customize page for the portlet. If the parameter you are exposing has a corresponding OracleAS Portal page parameter, and you leave the parameter value empty in the Customize page, the portlet inherits the page parameter's value. If the user enters a value for the report portlet's parameter, that value will override the page parameter value.

> **Note:** Running reports from within OracleAS Portal requires the HTML iframe tag, which is not supported in Netscape 4.x. As a result, the following limitations apply when using Netscape 4.x:
>
> ■ A report portlet cannot display in place if you are using HTTPS or if it is not a JSP report. You need to click on the portlet title to see the report in a separate browser window.
>
> ■ A report portlet cannot be scheduled to run via the Customize link if you are using HTTPS.

13.4.4 Adding the Reports Component as an Item to a Page

You can add an Oracle Reports component to a page as an item link using the Oracle Reports item type.

> **Note:** This item type must be included from the hidden list of
> item types and can be configured only if you are the page group
> administrator.

1. If you are not already on the Builder page, click **Builder** at the top of the page.

2. Click the **Build** tab.

3. In the Page Groups portlet, choose the name of the page group in which you want to place your report item link.

4. Create a new page by clicking **Create a Page** or edit an existing page by entering the name of an existing page and clicking **Edit**.

5. If you are creating a new page, follow the steps in the wizard and click the question mark in the upper right corner for additional information about the available settings. Click **Finish** when you are done.

6. If you are editing an existing page, skip to the next step.

7. Click the **Add Item** link. The Oracle Reports item type is available as a hidden item type. To include it as an available item type, click the **configure the list of available item types** link.

8. Select Oracle Reports in the Hidden Item Types list and click the **>** link to move it to the Visible Item Types list. Alternatively, you can click the **>>** link to move the entire Hidden Item Types list to the Visible Item Types list.

9. Click **OK**.

10. Select the Oracle Reports item type in the **Content Item Types** menu and click **Next**. The Add Oracle Reports page displays.

11. Enter a Display Name that users of your portal will view when clicking your report.

12. Select from the available default Oracle Reports components.

13. Select **Display Parameter Form** if you have require any user inputs before your report is displayed.

14. Select **Link That Displays Item In New Browser Window** to ensure that the report is viewed in a separate page.

15. Click **Finish**. The Oracle Reports item now displays as a link in your page.

16. Click the link to run the report and provide any parameters required, if **Display Parameter Form** is selected.

13.5 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.

URL syntax can be presented in the following forms:

- Full URL request, for example:

  ```
  http://your_webserver.domain_
  name:port/alias/rwservlet?report=myreport.rdf&userid=username/password@my_
  db&server=server_name&desformat=html&destype=cache
  ```

  If you require additional command line options, then refer to Command Line Options for a list of valid `rwclient` command line options.

- Simplified URL request using key mapping, for example:

  ```
  http://your_webserver.domain_name:port/alias/rwservlet?key1
  ```

13.6 Sending a Request to the URL Engine

If you have activated the Reports Server's URL engine, you can send job requests to the URL engine by using the following command line options:

- `urlParameter` identifies the URL to be placed in the cache. For example, `http://www.oracle.com` or a JSP report.

- `jobType` is the name of a job type (e.g., `urlEngine`) in the server configuration file that is associated with a URL engine.

  > **Note:** For information on activating the URL engine, refer to Configuring OracleAS Reports Services.

For example, a request that specifies an external URL for `urlParameter` might look like the following:

```
http://your_
webserver:portnum/reports/rwservlet?server=ReportsServer+jobType=urlEngine+urlPa
rameter="http://www.oracle.com"+destype=mail+desname=foo@bar.com+desformat=htmlc
ss
```

Alternatively, a request that specifies a JSP report for `urlParameter` would look like the following:

```
http://your_
webserver:portnum/reports/rwservlet?server=ReportsServer+jobType=rwurl+destype=c
ache+urlParameter="http%3A%2F%2Flocalhost%2Ffoo.jsp%3Fuserid%3Dscott%2Ftiger@ora
DB%3Fserver%3DreportsServer"
```

> **Note:** If the URL has special characters, they must be encoded as per the x-www-form-urlencoded format.

13.7 Running Reports through a Web Service

In many cases, reports are integrated components of some larger application rather a stand alone application themselves. Hence, it can be useful to generate report requests from within an application. We accomplish this goal by exposing OracleAS Reports Services as a Web service. This Web service may then be called from within any Web service aware environment (e.g., a Java application). For example, suppose that you have a Java-based expense reporting form and you want to allow users to generate a PDF version of their expense reports from it each time that they complete an expense form in your system. By creating a Java proxy Oracle Reports Web Service, you could then easily reference it from your Java development environment (e.g., Oracle JDeveloper) and add a button that invokes OracleAS Reports Services to generate the PDF file.

> **See Also:** Using the Oracle Reports Web Service
>
> For more information on the Oracle Reports Web service and installing and using the sample proxy and Java client.

13.8 Scheduling Reports to Run Automatically

You can use the server to run reports automatically from Reports Queue Manager, OracleAS Portal, or with the SCHEDULE command line option. The scheduling feature enables you to specify a time and frequency for the report to run.

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

If you publish a report as a portal component on an OracleAS Portal page, then you can schedule the report request to run automatically and push the resulting reports to specified pages. Refer to *OracleAS Portal online help* for more information.

The SCHEDULE keyword is available for use with the `rwclient`, `rwservlet`, and `rwcgi` commands. See Command Line Options, SCHEDULE for more information.

13.9 Additional Parameters

When you send a request to the Reports Server, the following additional parameters, the values of which you cannot change, are implicitly passed along with your request:

Table 13–5   Additional parameters passed with a report request

Name	Description
ACCEPT_LANGUAGE	The comma separated list of languages accepted by the browser/user.
REMOTE_ADDR	The remote IP address from which the user is making the request.
REMOTE_HOST	The remote host name from which the user is making the request.
SCRIPT_NAME	The virtual path of the script being executed.
SERVER_NAME	The host name or IP address of the server on which the Reports Servlet is running.
SERVER_PORT	The port number of the server on which the Reports Servlet is running.
SERVER_PROTOCOL	The name and revision of the information protocol with which the request was sent.
USER_AGENT	The description of the remote client's browser.

13.10 Reusing Report Output from Cache

When you run a report, a copy of the report output is saved in the OracleAS Reports Services cache. Subsequently, if an identical report is run (that is, with the same cache key), then the current request is recognized as a duplicate job.

There are several scenarios where reports caching takes effect:

- When a new job request "A" comes to the Reports Server, and there is another job "B" that has the same cache key in the Current Jobs Queue (where it is waiting for an available engine or is in the middle of execution), then job "A" will use the output from job "B".

The job cache key excludes the `destype`, `desname`, `server`, and `tolerance` parameters, and includes almost all other parameters.

This level of cache happens automatically. You don't need to specify any other parameters in the command line for it to work.

- If the user specifies TOLERANCE=n (where n is a number in units of minutes) in the new job request "A", and it doesn't happen, then Reports Server will try to find a job in the Finished Jobs Queue which was successfully completed within n minutes. If Reports Server can find such a job, then the new job request "A" will return the output of job "B".

> **Note:** Refer to Command Line Options for more information about the TOLERANCE command line option.

- In a clustered environment, duplicate job checking (i.e., jobs with the same cache key) is executed across cluster members. If a duplicate job is found in another server in the same cluster, the job request will be transferred to that server to retrieve the cached result.

OracleAS Reports Services cache results are persistent. If the Reports Server is shut down, once it is up again all the previous cache results are recovered and ready to use again.

13.10.1 Usage Notes

- You can set the cache size through Reports Queue Manager or via the `cache` element in the server configuration file (`server_name.conf`). Reports Server attempts to keep the total size of cache files below the set limit, deleting the oldest cache files. In addition, you can empty the cache through Reports Queue Manager.

For more information on setting the cache, refer to the *Reports Queue Manager online help*, and see Configuring OracleAS Reports Services.

13.11 Using a Key Map File

If you choose to provide users with a URL or add a hyperlink to a Web site, then you can use a key map file to simplify or hide parameters in your URL requests.

The key map file contains command strings for running reports, each headed by a unique key identifier. Except when you run a report as a JSP, you reference only this

key in the runtime URL. The server or servlet sends the key value to the map file (cgicmd.dat), which in turn returns the command associated with the specified key to the server or servlet for processing. By using key mapping, the command line options are all hidden from the user.

Key mapping is useful for:

- Shortening the URL, making it more convenient to use

- Remapping the runtime commands without having to change the original URL

- Standardizing several typical run configurations for your company

- Hiding certain parameters from users (for example, the database connect string)

- Restricting the parameters users can use to run a report

When you specify a key name from the key map file (cgicmd.dat), it must always be at the beginning of the query string (after the question mark) in a report request URL. An exception to this is if you use the cmdkey command line keyword, and express the key name as its value: cmdkey=*keyname*. In this case, you can place the key name anywhere in the query string within the report request URL. The cmdkey keyword can be used with jobs run as JSPs and with the rwservlet command.

> **Note:** You'll find more information about the cmdkey keyword in Command Line Options.

13.11.1 Enabling Key Mapping

Key mapping is enabled when any of these conditions are met:

- A valid file with the standard file name, cgicmd.dat, is present in the default location: the *ORACLE_HOME*\reports\conf\ directory on the Web server machine (on either Windows or UNIX).

- A valid key map file is entered in the Reports Servlet configuration file (rwservlet.properties) under the KEYMAPFILE parameter.

- When rwcgi is used, when the REPORTS_CGIMAP environment variable on the Web server machine specifies the name of a valid key map file. See Reports-Related Environment Variables for more information.

13.11.2 Adding Key Mapping Entries to a Key Map File

To add key mapping entries to a key map file:

1. Navigate to the `cgicmd.dat` file on the machine that hosts your Reports Server, and open it with a text editor.

   You'll find this file in the following directory on both Windows and UNIX:

   ```
   ORACLE_HOME\reports\server\conf\cgicmd.dat
   ```

2. Add a key mapping entry. For example:

   ```
   key1: report=your_report.rdf userid=username/password@my_db desformat=html
   SERVER=server_name.cluster_name (if present)> destype=cache
   ```

   In this example, `key1` is the name of the key.

   Except for the special parameters that are described in the file itself, the command line options follow the syntax rules of `rwclient`. See Command Line Options for more information.

3. Add or update the hyperlinks on your Web page.

   For more information, see Specifying a Report Request from a Web Browser.

13.11.3 Using a Key with Everything but JSPs

When you place a key name in a report request URL, it must always be the first value within the query string (immediately after the question mark). For example:

```
http://…/rwservlet?keyname
```

Below is an example of a key mapping for a restricted run with a parameter form.

The URL might be:

```
http://web_server.domain_name:port/cgi-bin/rwcgi.exe?key&par1&par2&parN
```

The key mapping file might contain:

```
KEY: REPORT=myreport DEPTNO=%1 MYPARAM=%2 %*
```

This would generate the equivalent of the following command line request:

```
rwclient REPORT=myreport DEPTNO=par1 MYPARAM=par2 parN
```

Usage Notes

- In `rwcgi` URLs, the first option (that is the first information after the question mark) is treated as a key if it is not otherwise a part of a name/value pair. If the

first option is not a name/value pair (i.e., keyword=*value*), then the whole command line is used in lieu of a cgicmd.dat key entry.

13.11.4 Using a Key with a Report Run as a JSP

When you run a report as a JSP and want to call a command key in the cgicmd.dat file, you must use the cmdkey keyword in your URL. For example, your JSP URL might look like this:

```
http://…/myreport.jsp?cmdkey=key
```

> **Note:** You can also use cmdkey with the rwservlet command.

When you use cmdkey with a JSP or rwservlet, you can place it anywhere within the query string. For example:

```
http://…/example.jsp?parameter1=value1&cmdkey=keyname
http://…/rwservlet?parameter1=value1&cmdkey=keyname
```

Usage Notes

■   When using key mapping, the order in which the parameters are substituted from the URL into the key is determined by the placement of CMDKEY in the URL. For example, suppose you have a key such as the following one in the cgicmd.dat file:

```
mykeys: DEPTNO=%1 MYPARAM=%2
```

Now, you execute a JSP report that references this key as follows:

```
http://neptune.world.com:80/jsp/myreport.jsp?userid=scott/tiger@hrdb
  &cmdkey=mykeys&10&test
```

Because of the placement of CMDKEY in this URL, the 10 corresponds to %1 and test corresponds to %2. Even though they are not the first and second parameters in the URL, 10 and test are the first and second parameters to follow CMDKEY in the URL.

14

Using the Oracle Reports Web Service

A Web service is an application that is built on standard Internet and XML technologies and has the following characteristics :

- Public interfaces and bindings defined and described using XML
- Publishes these public interfaces and bindings across the network for use by other programs

A Web service accepts a request, performs its function based on the request, and returns a response. The request and the response can be part of the same operation, or they can occur separately, in which case the consumer does not need to wait for a response. Both the request and the response usually take the form of XML, a portable data-interchange format, and are delivered over a wire protocol, such as HTTP.

Web service transactions are usually conducted between businesses. A business that is a provider of one service can also be a consumer of another service. A Web service consumer can also be a client device, such as a thin client connecting to the Web service provider over a lightweight protocol.

This chapter discusses the Oracle Reports Web service and contains the following sections:

- Overview
- Getting Started
- Installing and Using the Sample Proxy and Java Client

14.1 Overview

Oracle Reports provides several ways of submitting a job request to the server-infrastructure for processing:

- `rwservlet`

  `rwservlet` translates and delivers a job request between HTTP and the Reports Server, such as when submitting from a Web browser or via the event-driven publishing API.

- `rwcgi`

  `rwcgi` translates and delivers a job request between HTTP and the Reports Server, such as when submitting from a Web browser or via the event-driven publishing API. `rwcgi` is maintained only for backward compatibility.

- `rwclient`

  `rwclient` parses and transfers a command line to run a a report on a remote Reports Server.

- Oracle Forms

  Oracle Forms is a rapid application development (RAD) tool, used to build highly scalable Internet database applications.

Integrating the Oracle Reports technology into custom applications, especially JAVA applications, require the implementation of the mechanisms used by `rwservlet`, `rwcgi`, `rwclient`, and Oracle Forms to submit jobs to the server from within those applications.

The `RWWebService` servlet provides the necessary public interfaces and bindings — required to be exposed and to function as a Web service. This functionality enables any application developer to include the Oracle Reports in their application.

14.2 Getting Started

This section outlines the steps necessary for :

- Invoking the rwwebservice servlet
- Viewing the WSDL

14.2.1 Invoking the rwwebservice servlet

To invoke the `RWWebService` servlet:

1. Start an Oracle Application Server Containers for J2EE (OC4J) instance where the Oracle Reports instance resides.

2. Enter the following URL in the address field of your browser:
   `http://`*`host:port`*`/reports/rwwebservice`

   This takes you to the `RWWebService` endpoint. The `RWWebService` endpoint page enables you to do the following:

   a. View the Oracle Reports Web service WSDL.

   b. Run any `RWWebService` command using a Web based UI.

   c. Download the proxy JAR files and proxy sources to invoke the Oracle Reports Web service using the sample Java client.

14.2.2 Viewing the WSDL

The Web Service Description Language (WSDL) is an XML format for describing available services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

Note: Oracle Reports Web service does not support dynamic discovery of the WSDL by publishing to the universal description, discovery, and integration (UDDI) server.

---

1. Click the **Service Description** link on the `RWWebService` Web page to view the Oracle Reports Web service's WSDL document.

Note: Use Internet Explorer to view the WSDL XML output. When you use Netscape (7.2 and above) you must save the page as a `.xml` file and use Internet Explorer to open the file, for example, `rwwebservice.xml`.

---

2. The last entry in the WSDL is the service description and contains the location of the WebService:

   ```
   <soap:address location="http://localhost:8888/reports/rwwebservice" />
   ```

Figure 14–1   Viewing the WSDL

Ensure that the URL and port number,
`http://hostname:portnumber/reports/rwwebservice`, defined is
correct.

> **Note:**   The hostname specified should be the hostname where the
> OC4J instance is running and not where the Reports Server is
> running.

If the URL is not correct, you must do the following:

a.   Shutdown OC4J.

b.   Delete the `__java_stateless_rpc` located under the *ORACLE_
HOME*`\j2ee\OC4J_INSTANCE_
NAME\application-deployments\reports\web\temp\` directory.

c.   Restart OC4J.

d. Verify that the URL defined reflects:
`http://`*hostname*`:`*portnumber*`/reports/rwwebservice.`

Oracle Reports WSDL

```
 <?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="RWWebService"
targetNamespace="http://oracle.reports.rwclient/RWWebService.wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://oracle.reports.rwclient/RWWebService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <documentation>WSDL for Service: RWWebService, generated by Oracle WSDL
toolkit (version: 1.1)</documentation>
- <types>
  <schema targetNamespace="http://oracle.reports.rwclient/RWWebService.xsd"
xmlns:tns="http://oracle.reports.rwclient/RWWebService.xsd"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
  </types>
- <message name="runJobInput">
  <part name="param0" type="xsd:string" />
  <part name="param1" type="xsd:boolean" />
  </message>
- <message name="getServerInfoOutput">
  <part name="output" type="xsd:string" />
  </message>
  <message name="getAPIVersionInput" />
- <message name="getAPIVersionOutput">
  <part name="output" type="xsd:string" />
  </message>
...
- <portType name="RWWebServicePortType">
- <operation name="getServerInfo">
  <input message="tns:getServerInfoInput" />
  <output message="tns:getServerInfoOutput" />
  </operation>
- <operation name="getJobInfo">
  <input message="tns:getJobInfoInput" />
  <output message="tns:getJobInfoOutput" />
  </operation>
...
- <service name="RWWebService">
- <port name="RWWebServicePort" binding="tns:RWWebServiceBinding">
  <soap:address location="http://localhost:8888/reports/rwwebservice" />
```

```
</port>
</service>
</definitions>
```

14.2.3 Oracle Reports Web Service Operations

Oracle Reports exposes the `RWWebService` servlet as a Web service with its public interfaces and bindings defined and described using XML. These public interfaces and bindings are published across the network through the WSDL.

The various operations supported by the `RWWebService` endpoint are:

- getAPIVersion
- getServerInfo
- getJobInfo
- killJob
- runJob

14.2.3.1 getAPIVersion

The `getAPIVersion()` operation returns the version details of the Reports Server in XML format. This operation takes no parameters.

> **Note:** `getAPIVersion` is the only operation that returns the entire SOAP response along with the result (in a string). The other operations, e.g., `runJob` return the response as an XML block embedded within the SOAP response.

To view the `getAPIVersion` response:

1. Click the `getAPIVersion` link. The Test page should display no parameters and include only a **Invoke** button to submit the request.

2. Click **Invoke**. The SOAP response is displayed in a new window.

The following is a sample response of a `getAPIVersion` operation:

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <SOAP-ENV:Body>
```

```
- <ns1:getAPIVersionResponse
xmlns:ns1="http://oracle.reports.rwclient/RWWebService.wsdl"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <return xsi:type="xsd:string">9.0.4.0.9</return>
  </ns1:getAPIVersionResponse>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

14.2.3.2 getServerInfo

The `getServerInfo(String serverName, String authId)` operation takes two parameters and returns the Reports Server information in an XML format.

The valid parameters are :

■ *serverName*: A valid non-null server name. This operation returns an error if the specified server is not running in the network.

■ *authId*: A string in the form of username/password, must be specified for a secured server. This parameter is ignored for a non-secure server.

To view the `getServerInfo` response:

1. Click the `getServerInfo` link. The Test page should display the relevant parameter fields and an **Invoke** button to submit the request.

2. Enter the Reports Server name (`param0`) and `authid`(`param1`).

3. Click **Invoke**. The SOAP response is displayed in a new window.

The following is a sample output of the `getServerInfo` operation:

```
<?xml version = '1.0' encoding = 'ISO-8859-1' standalone = 'yes'?>
<serverInfo name="repserv" version="9.0.4.0.7">
   <host>incq246bc</host>
   <processId>2588</processId>
   <startTime>27-May-2003 10:09:34</startTime>
   <queue maxQueueSize="1000"/>
   <engine id="rwEng" activeEngine="1" runningEngine="0"/>
   <engine id="rwURLEng" activeEngine="1" runningEngine="0"/>
   <performance>
      <property name="successfulJobs" value="6"/>
      <property name="currentJobs" value="0"/>
      <property name="futureJobs" value="0"/>
      <property name="transferredJobs" value="0"/>
      <property name="failedJobs" value="0"/>
      <property name="responseTime" value="2124"/>
   </performance>
```

</serverInfo>

14.2.3.3 getJobInfo

The getJobInfo(Integer jobId, String serverName, String authId) operation returns the job information in XML format.

The valid parameters are:

- *jobId:* JobId of the job for which information is required.

- *serverName*: A valid non-null Server name value must be supplied. This operation returns an error if the specified server is not running in the network.

- *authId*: A string in the form of username/password, must be specified for a secured server. For a non-secure server this parameter is ignored.

To view the getJobInfo response:

1. Click the getJobInfo link. The Test page should display the relevant parameter fields and an **Invoke** button to submit the request.

2. Enter the jobID (param0), Reports Server name (param1), and authid(param2).

3. Click **Invoke**. The SOAP response is displayed in a new window.

The following is a sample output of a getJobInfo operation for job id=3:

```
<?xml version = '1.0' encoding = 'ISO-8859-1' standalone = 'yes'?>
<serverQueues>
   <job id="3" queueType="past">
      <name>test.rdf</name>
      <type>report</type>
      <status code="4">Finished successfully</status>
      <owner>RWUser</owner>
      <server>repserv</server>
      <destination>
         <desType>cache</desType>
         <desFormat>html</desFormat>
         <file>21748116.htm</file>
         <file>217481161.jpg</file>
         <file>217481160.jpg</file>
      </destination>
      <timingInfo>
         <queued>27-May-2003 10:21:50</queued>
         <started>27-May-2003 10:21:50</started>
         <finished>27-May-2003 10:21:51</finished>
```

```
            </timingInfo>
        </job>
</serverQueues>
```

14.2.3.4 killJob

The killJob(Integer jobId, String serverName, String authId) operation kills the job based on the job id specified and returns the status of the operation in XML format.

The valid parameters are:

- *jobId:* JobId of the job for which information is required.

- *serverName*: A valid non-null Server name value must be supplied. This operation returns an error if the specified server is not running in the network.

- *authId*: A string in the form of username/password, must be specified for a secured server. For a non-secure server this parameter is ignored.

To view the killJob response:

1. Click the killJob link. The Test page should display the relevant parameter fields and an **Invoke** button to submit the request.

2. Enter the jobID (param0), Reports Server name (param1), and authid(param2).

3. Click **Invoke**. The SOAP response is displayed in a new window.

The following is a sample output of a killJob operation for Job ID=3:

```
<?xml version = '1.0' encoding = 'ISO-8859-1' standalone = 'yes'?>
<serverQueues>
   <job id="3" queueType="past">
       <name>test.rdf</name>
       <type>report</type>
       <status code="7">Canceled upon user request</status>
       <owner>RWUser</owner>
       <server>repserv</server>
       <destination>
          <desType>cache</desType>
          <desFormat>html</desFormat>
       </destination>
       <timingInfo>
          <queued>27-May-2003 10:21:50</queued>
          <started>27-May-2003 10:21:50</started>
          <finished>27-May-2003 10:22:00</finished>
```

```
          </timingInfo>
       </job>
</serverQueues>
```

14.2.3.5 runJob

The `runJob(String commandLine, Boolean synchronous)` operation runs a job to the Reports Server specified as part of the `commandLine` parameter.

> **Note:** Oracle Reports Web service does not return the job output or the actual report.

The valid parameters are:

- *commandLine*: The complete command line syntax for submitting a job. For example:

  ```
  server=repserv report=test.rdf destype=file desname=output.pdf desformat=pdf
  userid=scott/tiger@v815
  ```

- *synchronous*: A Boolean object to indicate if the job should be run synchronously.

To view the `runJob` response:

1. Click the `runJob` link. The Test page should display the relevant parameter fields and an **Invoke** button to submit the request.

2. Enter the command line syntax (`param0`), whether the job should run synchronously (T/F, Y/N) (`param1`).

3. Click **Invoke**. The SOAP response is displayed in a new window.

The following is a sample output of a `runJob` operation:

```xml
<?xml version = '1.0' encoding = 'ISO-8859-1' standalone = 'yes'?>
<serverQueues>
   <job id="7" queueType="current">
      <name>test.rdf</name>
      <type>report</type>
      <status code="1">Waiting in the queue</status>
      <owner>RWUser</owner>
      <server>repserv</server>
      <destination>
         <desType>file</desType>
         <desName>output.pdf</desName>
```

```
            <desFormat>pdf</desFormat>
         </destination>
         <timingInfo>
            <queued>27-May-2003 10:22:00</queued>
            <started>27-May-2003 10:22:00</started>
            <finished>27-May-2003 10:22:00</finished>
         </timingInfo>
     </job>
</serverQueues>
```

14.3 Installing and Using the Sample Proxy and Java Client

The `RWWebService` Web page contains a link to a sample proxy. This sample proxy invokes the Web service internally using the appropriate SOAP messages. Thus, the proxy accesses the various operations performed by the Web service and invokes them using the appropriate parameters.

The following procedure outlines the necessary steps involved in installing the proxy:

1. Download the `rwwebservice.zip` file from the Proxy Jar link displayed on the RWWebService Web page.

2. Include the path to the `rwwebservice.zip` file in your system classpath.

3. Include the *ORACLE_HOME*\soap\lib\soap.jar, *ORACLE_HOME*\j2ee\home\lib\http_client.jar; entries in your system classpath.

4. Include either one of the following in your system classpath:

- The `xmlparserv2.jar` located in the *ORACLE_HOME*\lib\xmlparserv2.jar directory.

- The Xerces 1.4.4 parser, `xerces.jar`.

   > **Note:** You must specify the location of the xmlparser in the system classpath. If you do not specify the location, the SOAP response will be displayed minus the <> symbols.

5. Modify the `RWWebServiceTest.java` to reflect your Reports Server name and the `authid`. For more information on constructing a Java client, refer to Example 14–1.

> **Note:** The `authid` for non-secured Reports Server is `null`.

6. Compile and run the `RWWebServiceTest.java` file.

7. Run the various operations using the sample Java client. For example, get the API Version (getAPIVersion), run a job(runJob), check the status of any job (getJobInfo), or get the Reports Server information (getServerInfo).

> **Note:** You can submit many jobs concurrently from multiple windows using the sample Java client.

Example 14–1 illustrates the contents of the `RWWebServiceTest.java` file.

Example 14–1 RWWebServiceTest.java

```
/*
 * $Id: RWWebServiceTest.java
 * @author Anil Sharma
 *
 * Copyright (c) Oracle Corporation 2003.  All Rights Reserved
 *
 * FUNCTION
 * This is a sample class to demonstrate how the Oracle Reports WebService
 * Proxy class(oracle.reports.rwclient.proxy.RWWebServiceProxy) can be used
 * to invoke the Reports WebService from Java Clients. Java based Reports
 * Clients will use the demonstrated mechanism for invoking & parsing the
 * results using
 *
 * NOTES
 * 'oracle.reports.rwclient.proxy.RWWebServiceProxy' class is supplied as part
 * of rwwebservice.zip file which can be downloaded by invoking the Reports
 * WebService from a browser. Please consult Chapter 14 of Oracle Application
Server Reports Services Publishing Reports to the Web manual, available on
 * the Oracle Technology Network Oracle Reports Documentation page
 * (http://otn.oracle.com/docs/products/reports/content.html) for details.
 *
 * CREATED   Anil Sharma    06/13/03
 */


import oracle.reports.rwclient.proxy.RWWebServiceProxy;
```

```java
/**
 * This class creates an instance of RWWebServiceProxy and makes API
 * calls on it to interact with Reports Server WebService. The result from
 * the webservice is usually an XML object which is printed as-is to the
 * Standdard output stream. Java based Reports Clients making use of this
 * WebService class might need to parse the XML to extract meaningful
 * information.
 *
 */

public class RWWebServiceTest
{

  public static void main(String[] args)
  {

    String serverName = "repserv";          //Name of the Reports Server
    String authid     = "portal/welcome1";  //authid, should be null if
                                            //server is not secured.
    String cmdline    = "server=repserv report=test.rdf "+
          "destype=file desname=output.pdf desformat=pdf "+
          "userid=scott/tiger@orcl";
    try
    {
      RWWebServiceProxy proxy = new RWWebServiceProxy();

      /**
       * The following piece of code invokes proxy class' getAPIVersion()
       * to get the Reports Server version information.
       */
      System.out.println("Get Reports Server Version:");
      System.out.println("RESULT:"+proxy.getAPIVersion());

      /**
       * Get the Reports Server Information in XML format. This will contain
       * some server runtime as well as configuration information.
       */
      System.out.println("Get Server Info:");
      System.out.println("RESULT:\n"+proxy.getServerInfo(serverName, authid));

      /**
       * Get information about a particular job (the job ID needs to be
       * specified.
       */
      System.out.println("Get JobInfo for Job Id=3:");;
```

```
            System.out.println("RESULT:\n"+proxy.getJobInfo(new Integer(3),
    serverName, authid));

         /**
          * Kill a job with a given job ID.
          */
         System.out.println("Kill job with Job Id=3:");
         System.out.println("RESULT:\n"+proxy.killJob(new Integer(3), serverName,
    authid));

         /**
          * Submit a job to the server. The command string takes the same form
          * as the one used for rwclient or any other Oracle Reports client.
          * You can specify whether to run the job synchronously or not. The
          * returned string is in XML format indicating the job status. Please
          * note that with Oracle Reports version 10g (9.0.4), you can not get the
          * job output.
          */
         System.out.println("Run a job on server");
         System.out.println("RESULT:\n"+proxy.runJob(cmdline, new Boolean(true)));

      }
      catch (Exception e)
      {
        e.printStackTrace();
      }
    }
}
```

15

Creating Advanced Distributions

When you wish to define an advanced distribution for your report, you can design the distribution by developing a distribution XML file. This file can specify which section or sections of a report should go to what destination via what format of output. In one distribution XML file, you can specify many different destinations, including custom (pluggable) destinations you design.

This chapter provides information on creating a distribution XML file and some example use cases. It includes the following main sections:

- Distribution Overview
- Introduction to Distribution XML Files
- Elements of a Distribution XML File
- Distribution XML File Examples
- Defining Custom/Pluggable Destinations

15.1 Distribution Overview

Although distribution XML files are not required for specifying the distribution of report output, they are useful for complex distributions. For example, there may be times when you want to publish the output of one report in a variety of ways. You might want to send an executive summary of a report to senior management while mailing detailed breakdowns to individual managers. In this case, you might produce a single report with two report sections: a portrait-sized summary section and a landscape-sized detail section. You would associate the detail section with a data model group that lists the managers, then alter the destination on each instance of the group to send each department's output to its related manager.

The distribution XML file tames distribution complexity by enabling you to define multiple outputs for a given report in one XML file, then call that file from a command line or URL.

15.2 Introduction to Distribution XML Files

15.2.1 The distribution.dtd File

When you create a distribution XML file, you follow the syntax defined in the `distribution.dtd` file located in the following directory (Windows and UNIX use the same path):

ORACLE_HOME`\reports\dtd`

As you look through the following sections, it may be useful to you to print the distribution.dtd file and refer to it as various elements and attributes are described.

> **Note:** information provided in distribution XML file is case sensitive. The user must preserve case of various elements and attributes as specified in the `distribution.dtd` file.

The `distribution.dtd` file lists all elements that are valid within a distribution XML file. Each of these elements have attributes. Attributes that come with default values need not be specified, unless you wish to override the default.

You can create a dynamic distribution by introducing variable values into many different attributes. Variable values reference columns that are present in the report that is using the distribution XML file.

15.2.2 Using Variables Within Attributes

You can use variables within attributes by entering `&column_name` or `&<column_name>` in the place of a static value.

> **Note:** The ampersand (&) and less-than symbol (<) have specific meanings in XML, but they are also required symbols for certain Oracle Reports Developer command line options (for example, lexical parameters require the ampersand symbol). To avoid conflict with the XML meanings of these symbols when you set up variables, specify the encoded version of the ampersand (`&`) and less-than and greater-than symbols (`<` and `>`). For example:
>
> Here is what the variable looks like *improperly* coded in an XML file:
>
> ```
> <mail id="a1" to="&<manager>@mycompany.com" …
> ```
>
> Here is what the variable looks like *properly* coded in an XML file:
>
> ```
> <mail id="a1" to="&amp;&lt;manager&gt;@mycompany.com" …>
> ```
>
> There is no special requirement for the greater-than symbol (>) used with variables, but for consistency, we recommend that you use the encoded version (>).

The variable syntax you use depends on whether the value is expressed by itself or in combination with other values or strings. For example, a value for a "to" attribute in a mail element might be expressed as either:

```
<mail id="a2" to="&amp;email" …>
```

OR

```
<mail id="a3" to="&amp;&lt;first_name&gt;.&amp;&lt;last_name&gt;@myco.com …>
```

In the first example (id="a2"), the variable's referenced column (`email`) contains a full e-mail address and does not require additional information. The second example (id="a3") uses a combination of variable values (`first_name` and `last_name`) and static text to construct an e-mail address (static text is the period after first_name and *@myco.com*). In both cases, you will get dynamic e-mail addressing. The example you use will depend on whether the variable contains all the information you need or requires additional information in order to be complete.

For even more complex layouts, you can also reference report columns you created with PL/SQL formulas. For example, in your report you may define the PL/SQL column:

```
PL/SQL formula CF_MAILID: return(:first_name||'.'||:last_name)
```

You'd reference this column in the distribution XML file as:

```
to="&amp;&lt;CF_MAILID&gt;@mycompany.com"
```

15.3 Elements of a Distribution XML File

The elements of a distribution XML file include:

- destinations
- foreach
- mail
- body
- include
- file
- printer
- destype
- attach
- property

Most of these elements have attributes that define the behavior of the element. The following sections describe the distribution XML file elements and their associated attributes. Distribution XML File Examples provides use cases that demonstrate the distribution XML file elements and attributes in action.

15.3.1 destinations

Example
```
<destinations>
[One or more distribution specifications]
</destinations>
```

Required/Optional
Required. You must have no more or less than one destinations element in your distribution XML file.

Description

The `destinations` element opens and closes the content area of the distribution XML file. In terms of the distribution XML file's tagging hierarchy, all the other elements are subordinate to the `destinations` element.

The `destinations` element has the following sub-elements:

- foreach
- mail
- file
- printer
- destype

Each of these is discussed in the following subsections.

15.3.2 foreach

Example

```
<foreach>
    <mail id="a1" to="my_addressee@mycompany.com" subject="Fourth Quarter
    Results">
      <attach format="pdf" name="dept_&amp;department_ID&gt;.pdf"
      srcType="report" instance="this">
         <include src="mainSection"/>
      </attach>
    </mail>
</foreach>
```

OR

```
<mail id="a4" to="recipient@mycompany.com" subject="Regional Results">
    <foreach>
        <attach format="pdf" name="report.pdf" srcType="report" instance="all">
           <include src="mainSection"/>
        </attach>
    </foreach>
</mail>
```

Required/Optional

Optional. You can have as many `foreach` elements as you require.

Description

Use the `foreach` element to burst your distribution against a repeating group. You can use `foreach` only when the associated report definition file (either RDF, JSP, or XML) has its "Repeat On" property for the section that will be burst set to an appropriate group. The `foreach` element specifies that the distribution defined between its open and close parameters should be performed for each repeating group.

The Repeat On property can be set for a report section (Header, Main, and Trailer) to associate a data model break group to a section. By setting the Repeat On property for a section, you can generate multiple instances of a section, or a *repeating section*.

When you implement bursting and distribution in a report, you can generate *section-level distribution* by setting the Repeat On property for a section to a data model break group, which generates an instance of the section for each column record of that break group. Then, you can distribute each instance of the section as appropriate (for example, to individual managers in the MANAGER group).

If you set the Repeat On property for more than one of the Header, Main, and Trailer sections of a report, all Repeat On property values must be set to the **same data model break group**. If the Repeat On property for any one of the Header, Main, and Trailer sections is set to a different data model break group, Oracle Reports raises any of the following messages:

```
REP-0069: Internal Error
REP-57054: In-Process job terminated: Terminated with error
REP-594: No report output generated.
```

The `foreach` element has the following sub-elements:

- mail

- file

- printer

- destype

- attach

Each of these is discussed in the following subsections.

You can also use the `foreach` element as a sub-element of the `mail` element, as depicted in the second example provided at the start of this section. (In this example, assuming that `mainSection` repeats on G_DEPARTMENT_ID, the

example will produce a single attachment with all the instances of the report's mainSection in a single file.)

The foreach element works closely with the instance attribute of the attach and file elements. While foreach specifies that the distribution should be performed according to record groups, instance specifies whether the burst groups should be distributed in one file (instance="all") or distributed as separate files: one file for each group instance (instance="this").

When used with the mail element, foreach can mean different things according to its position relative to the mail element:

- When foreach precedes the mail element and instance="this", each group instance is dispatched as a separate mail. For example:

```
<foreach>
  <mail id="a1" to="managers@mycompany.com" subject="results">
  <attach name="department_&amp;&lt;department_id&gt;.pdf" instance="this">
  <include src="mainSection" />
  </attach>
  </mail>
</foreach>
```

  If the report is grouped according to department_id, and there are four departments, then there are four group instances. This means four e-mails per recipient, each e-mail with its own group instance attached: one e-mail has department 10's report attached; another e-mail has department 20's report attached; and so on. Each recipient receives all four e-mails.

- When foreach follows the mail element and instance="this", each group instance is attached to one e-mail going to each recipient. For example:

```
<mail id="a1" to="managers@mycompany.com" subject="results">
  <foreach>
    <attach name="department_&amp;&lt;department_id&gt;.pdf"
    instance="this">
    <include src="mainSection" />
    </attach>
  </foreach>
</mail>
```

15.3.3 mail

Example

```
<mail id="a1" to="jsmith@foo.com" subject="Results">
   <body srcType="text">
     Attached are quarterly results.
   </body>
   <attach srcType="report">
      <include src="headerSection"/>
      <include src="mainSection"/>
   </attach>
</mail>
```

OR

```
<mail id="a4" to="recipient@mycompany.com" subject="Regional Results">
   <foreach>
     <attach format="pdf" name="report.pdf" srcType="report" instance="this">
        <include src="mainSection"/>
     </attach>
   </foreach>
</mail>
```

Required/Optional

Optional. You can have as many mail elements as you require.

Description

Use the mail element to specify distributions via an outgoing SMTP-based mail server. Use it to specify the recipients, the subject, and the priority of the e-mail.

The mail element has three sub-elements:

- body

- attach

- foreach

Between an open and close mail element, there can be only one body sub-element and anywhere from zero to multiple attach and foreach sub-elements.

The mail element also has a set of related attributes. These are expressed within the mail tag. For example, the id, to, and subject attributes are expressed:

```
<mail id="a1" to="jsmith@foo.com" subject="Recent Hires">
```

Table 15–1 lists and describes the attributes associated with the mail element.

Table 15–1   Attributes of the mail element

Attribute	Valid values	Description
id	string	Required. A keyword, unique within a given distribution XML file, that identifies a particular mail element. This can be a combination of a text string and one or more numbers, for example id="a1". The id value must always start with an alpha character.
to	string	Required. Variable values allowed. The recipient(s) of the e-mail. Contains the full, formal e-mail address, for example:
		to="jsmith@foo.com"
		Multiple recipients must be separated with commas.
		Can also contain variable values that reference columns used in the associated report. See Section 15.2.2 for more information.
cc	string	Optional. Variable values allowed. The recipient(s) to receive a copy of the e-mail.
bcc	string	Optional. Variable values allowed. The recipient(s) to receive a blind copy of the e-mail.
from	string	Optional. Variable values allowed. The sender of the e-mail.
replyTo	string	Optional. Variable values allowed. The e-mail account where replies should be sent.
subject	string	**Default:** Mail Sent from &Report
		Optional. Variable values allowed. The subject of the e-mail. In the absence of a specified subject, the subject line will read: Mail Sent from [*Name of Report*]
priority	highest|high|normal|low|lowest	**Default:** normal
		The e-mail's delivery priority.

Table 15–1   Attributes of the mail element

Attribute	Valid values	Description
returnRecei pt	true|false	**Default:** `false` Indication of whether the replyto individual or account should be notified when the e-mail is received.
organizatio n	string	Optional. Variable values allowed. The name of the organization distributing the e-mail, for example: organization="Region 10 Sales" Or organization="&department_name"

> **Note:**   For the mail element to work properly, the Reports Server must know which outgoing SMTP mail server to send mail to. You specify this information in the Reports Server configuration file (*server_name*.conf). This file has a pluginParam element where you can enter the name of a mail server. For example:
>
> `<pluginParam name=mailServer>smtp01.mycorp.com</pluginParam>`
>
> For more information, see Configuring OracleAS Reports Services.

15.3.4 body

Example

On Windows
```
<mail id="a1" to="jsmith@foo.com" subject="Results">
   <body srcType="file">
      <include src="c:\mail\body.html"/>
   </body>
</mail>
```

On UNIX
```
<mail id="a1" to="jsmith@foo.com" subject="Results">
   <body srcType="file">
      <include src="/mail/body.html"/>
   </body>
```

```
</mail>
```

Required/Optional

Optional. You can have a maximum of one body element associated with a given mail element.

Description

The `body` element acts as a sub-element to the `mail` element. It specifies the content (or `body`) of the e-mail. With `body`, you can type a text string between the open and close `body` tag or use an [include](#) sub-element to specify either an external file, a report, or a section of a report. For example:

```
<mail id="a1" to="jsmith@foo.com" subject="Results">
    <body srcType="text">
        Attached are quarterly results.
    </body>
    …
```

Or

```
<mail id="a1" to="jsmith@foo.com" subject="Results">
    <body srcType="file">
        <include src="d:/reports/admin/results.html"/>
    </body>
    …
```

Or

```
<mail id="a1" to="&amp;&lt;first_name&gt;.&amp;&lt;last_name&gt;@myco.com"
subject="Quarterly Results">
    <body srcType="report" format="html">
      <include src="headerSection"/>
    </body>
    …
```

`Body` has three attributes: `srcType`, `format`, and `instance`. They are described in [Table 15–2](#).

Table 15–2   Attributes of the body sub-element of mail

Attribute	Valid values	Description
srcType	file|report|text	**Default:** report
		The source for content of an e-mail. The content is displayed in the body of the e-mail. In the absence of a specified srcType, the default is used.
format	html|htmlcss|ascii	**Default:** html
		Required when srcType is report with a format other than html, the default; otherwise format is optional. The format of the content.
instance	this|all	**Default:** all
		Used when the foreach element is also present. With a grouped report that is burst into separate reports, instance specifies whether the groups will be broken into separate content according to each group instance (this) or all contained within the same content (all).

15.3.5 attach

Example

```
<mail id="a1" to="jsmith@foo.com" subject="Results">
    <body srcType="text">
       Attached are quarterly results.
    </body>
    <foreach>
        <attach format="html" name="contacts.htm" srcType="report"
        instance="all">
            <include src="headerSection"/>
            <include src="mainSection"/>
        </attach>
    </foreach>
</mail>
```

Required/Optional

Optional. You can have as many attach elements as you require with a mail element. Note that attach is also a sub-element of foreach, and foreach requires that at least one of its sub-elements be used (out of mail, file, printer, destype, and attach).

Description

Attach specifies attachments to the e-mail. Additionally, attach must have at
least one include sub-element, and can have more than one if
srcType="report". Attach attributes are listed and described in Table 15–3.

Table 15–3   Attributes of the attach sub-element of mail

Attribute	Valid values	Description
format	pdf|html|htmlcss|rtf| ascii|xml|dflt	**Default:** pdf Required when srcType is report and the report format is other than pdf, the default; otherwise format is optional. The format of the attached material, for example format="htmlcss".
name	string	Optional. Variable values allowed. The filename of the attached material. Can also contain variable values that reference columns used in the associated report. See Section 15.2.2 for more information.
srcType	file|report|text	**Default:** report The source of the attachment, either a file, a report, or text.
instance	this|all	**Default:** all Used when the foreach element is also present. With a grouped report that is burst into separate reports, instance specifies whether the groups will be broken into separate content according to each group instance (this) or all contained within the same content (all).

Using these attributes in conjunction with the foreach element, you can design a
destination that will repeat on a group instance and generate an e-mail for each
group attachment. For example:

```
<foreach>
    <mail id="a2" to="first.name@myco.com,second.name@myco.com,
    third.name@myco.com, fourth.name@myco.com" subject="Department Summaries">
       <body srcType="text">
          Attached is the breakdown of department summaries for the last
          quarter.
       </body>
       <attach format="htmlcss" name="deptsum.html" srcType="report"
       instance="this">
```

```
            <include src="report"/>
        </attach>
      </mail>
</foreach>
```

By moving the location of the `foreach` element, you can generate one e-mail with multiple attachments: a separate one for each group instance.

```
<mail id="a2" to="first.name@myco.com,second.name@myco.com, third.name@myco.com,
fourth.name@myco.com" subject="Department Summaries">
    <body srcType="text">
        Attached is the breakdown of department summaries for the last
        quarter.
    </body>
    <foreach>
        <attach format="htmlcss" name="deptsum.html" srcType="report"
        instance="this">
            <include src="report"/>
        </attach>
    </foreach>
</mail>
```

15.3.6 include

Example

```
<mail id="a1" to="jsmith@foo.com" subject="Q4">
    <body srcType="text">
        Attached are quarterly results.
    </body>
    <attach srcType="report" format="pdf">
      <include src="report"/>
    </attach>
</mail>
```

Or

```
<mail id="a1" to="jsmith@foo.com" subject="Q4">
    <body srcType="text">
        Attached are quarterly results.
    </body>
    <attach srcType="report" format="htmlcss">
        <include src="headerSection"/>
    </attach>
</mail>
```

Or

```
<mail id="a1" to="jsmith@foo.com" subject="Q4">
    <body srcType="text">
       Attached are quarterly results.
    </body>
    <attach srcType="file">
        <include src="d:/management/reports/current/Q4.htm"/>
    </attach>
</mail>
```

Required/Optional

Required when used with `body` and `attach` when `srcType` is `report` or `file`, but not when `srcType` is `text`. Also required for `file`, `printer`, and `destype`. In the instances where it is required, you must have one and can have more than one `includes`.

Description

The `include` element is available for use with the body, attach, file, printer, and destype elements. It specifies the file, report, or report section to be included in the body of an e-mail, as an attachment to an e-mail, in the content of a file, in the printer output, or in the content of a custom destination type.

If you want to specify more than one section, but not the entire report, enter an `include` for each required section. For example:

```
<mail id="a1" to="jsmith@foo.com" subject="Results">
    <body srcType="text">
       Attached are quarterly results.
    </body>
    <attach srcType="report" format="htmlcss">
       <include src="headerSection"/>
       <include src="mainSection"/>
    </attach>
</mail>
```

If the preceding `body` or `attach` element has `srcType` of `file`, the subsequent `include` can specify the file either with a directory path and filename or with just the filename, provided the file is located in a directory listed in the `REPORTS_PATH` environment variable. For example:

```
<mail id="a1" to="jsmith@foo.com">
   <body srcType="file">
```

```
        <include src="q4sales.pdf"/>
    </body>
</mail>
```

If you do specify a path, use the appropriate standard for your platform. For example:

On Windows: `<include src="c:\management\reports\current\Q4.htm"/>`

On UNIX: `<include src="/management/reports/current/Q4.htm"/>`

No other XML elements are placed between an include element's open and close tags; however, include does have one attribute: src, described in Table 15–4.

Table 15–4   Attributes of the include sub-element when used with mail's body or attach

Attribute	Valid values	Description
src	*(path and) filename* report headerSection mainSection trailerSection	Required. The source of material specified in the preceding attach, body, file, printer, or destype element.
		Because the distribution XML file is called when you run a specific report, there is no need to specify the report's name or location in the src attribute when src="report".
		When the preceding body or attach element specifies a file srcType, provide the directory path and filename or just a filename, provided the file is located in a directory listed in the REPORTS_PATH environment variable.
		When the preceding body or attach element specifies a report srcType, specify the entire report (report) or provide the section(s) of the report to be included in the body or to be attached (e.g., headerSection, mainSection, and/or trailerSection).

15.3.7 file

Example

On Windows `<file id="a7" name="c:\management\reports\report.pdf"`
`format="pdf">`

```
      <include src="report"/>
</file>
```

On UNIX `<file id="a7" name="/management/reports/report.pdf" format="pdf">`
```
      <include src="report"/>
</file>
```

Or

```
<foreach>
    <file id="a7" name="section&amp;&lt;department_id&gt;.pdf" format="pdf"
    instance="this">
        <include src="mainSection"/>
    </file>
</foreach>
```

Required/Optional

Optional. You can have as many file elements as you require.

Description

Use the `file` element to specify distributions to a file. File elements have one sub-element: [include]. There must be at least one `include` sub-element and there may be more between an open and close `file` element.

When used with the [foreach] element and the instance="this" attribute, the `file` element can distribute each group instance of a grouped report to separate files. For example, if you group a report on department_id, and there are four departments, you can use the foreach/file/instance="this" combination to generate four files, each with a separate department's report. In this case, the `file` entry in the distribution XML file might look like this:

```
<foreach>
    <file id="a3" name="dept_&amp;&lt;department_id&gt;.pdf" format="pdf"
    instance="this">
        <include="report"/>
    </file>
</foreach>
```

In this example, all report sections (header, main, and trailer) must repeat on the same group instance (e.g., department_id).

File elements also have a set of related attributes. These are expressed within the file tag. For example, the "id" and "name" file attributes are expressed:

Windows: `<file id="a7" name="d:\reports\2001\q4sales.pdf">`

UNIX: `<file id="a7" name="/reports/2001/q4sales.pdf">`

Table 15–5 lists and describes the attributes associated with a `file` element.

Table 15–5 Attributes of the file element

Attribute	Valid values	Description
id	string	Required. A keyword, unique within a given distribution XML file, that identifies a particular file element. This can be a combination of a text string and one or more numbers, for example `id="a1"`. The id value must always start with an alpha character.
name	string	Required. Variable values allowed. The location and filename of the destination file. Enter a directory path. Include the filename. For example: **Windows:** `name="d:\reports\file.pdf"` **UNIX:** `name="reports/file.pdf` Can also contain variable values that reference columns used in the associated report. See Section 15.2.2 for more information.
format	`pdf|html|htmlcss|rtf|ascii|xml|bitmap`	**Default:** `pdf` The destination file format, for example: `format="htmlcss"`
instance	`this|all`	**Default:** `all` Used when the foreach element is also present. With a grouped report that is burst into separate reports, `instance` specifies whether the groups will be broken into separate files according to each group instance (this) or all contained within the same file (all).

15.3.8 printer

Example

On Windows
```
<printer id="a1" name="\\server_name\printer_name" copies="5">
    <include src="report"/>
```

```
</printer>
```

On UNIX

```
<printer id="a1" name="alias_to_registered_printer" copies="5" instance="all">
     <include src="report"/>
</printer>
```

Required/Optional

Optional. You can have as many printer elements as you require.

Description

Use the printer element to specify distributions to a printer. Printer elements have one sub-element: include. There must be at least one include sub-element and there may be more between an open and close printer element.

When used with the foreach element and the instance="this" attribute, the printer element can distribute each group instance of a grouped report to a separate print job. For example, if you group a report on department_id, and there are four departments, you can use the foreach/printer/instance="this" combination to generate four printed reports, each containing a separate department's report. In this case, the printer entry in the distribution XML file might look like this:

```
<foreach>
    <printer id="a7" name="\\server_name\printer_name" instance="this">
        <include="report"/>
    </printer>
</foreach>
```

In this example, all report sections (header, main, and trailer) must repeat on the same group instance (e.g., department_id).

Table 15–6 lists and describes the attributes associated with a printer element.

Table 15–6   Attributes of the printer element

Attribute	Valid values	Description
id	string	Required. A keyword, unique within a given distribution XML file, that identifies a particular file element. This can be a combination of a text string and one or more numbers, for example id="a1". The id value must always start with an alpha character.

Table 15–6 Attributes of the printer element

Attribute	Valid values	Description
name	string	Required. Variable values allowed. The destination printer. How you enter this information differs between Windows and UNIX.
		For Windows, specify the printer server name and the printer name. For example:
		name="\\server_name\printer_name"
		For UNIX, specify the alias assigned to a registered printer. For example:
		name="sales_printer"
		Can also contain variable values that reference columns used in the associated report. See Section 15.2.2 for more information.
copies	string	**Default:** 1
		Number of copies of each report or each report group instance to print.
instance	this|all	**Default:** all
		Used when the foreach element is also present. With a grouped report that is burst into separate reports, instance specifies whether the groups will be broken into separate printed reports according to each group instance (this) or all contained within the same printed report (all).

15.3.9 destype

Example

```
<destype id="acustom1" name="fax">
    <include src="headerSection"/>
    <property name="number" value="914925551212"/>
</destype>
```

Required/Optional

Optional. You can have as many destype elements as you require.

Description

Use the `destype` element to specify distribution to a custom destination, such as to a fax machine or an FTP site. You also use `destype` to specify distribution to a portal created with OracleAS Portal. The `destype` element allows for the use of two sub-elements: `property` and `include`. At least one include is required.

The inclusion of a custom destination type requires that you have a defined distribution handler in place to usher report content to the custom output destination.

> **Note:** Build a custom destination type via the OracleAS Reports Services Destinations API. Look for upcoming information about Reports APIs on the Oracle Technology Network, (`http://otn.oracle.com`).

When used with the `foreach` element and the `instance="this"` attribute, the `destype` element can distribute each group instance of a grouped report to a separate `destype` instance (e.g., a separate fax). For example, if you group a report on department_id, and there are four departments, you can use the `foreach/printer/instance="this"` combination to generate four `destype` instances, each containing a separate department's report. In this case, the `destype` entry in the distribution XML file might look like this:

```
<foreach>
    <destype id="a9" name="fax" instance="this">
       <include="report"/>
       <property name="number" value="&amp;&lt;fax_number&gt;"/>
    </destype>
</foreach>
```

In this example, all report sections (header, main, and trailer) must repeat on the same group instance (e.g., department_id).

Custom destination types also have a set of related attributes. These are expressed within the `destype` tag. For example, the "id", "name", and "instance" `destype` attributes are expressed:

```
<foreach>
    <destype id="a1" name="name_of_destination_type" instance="all">
</foreach>
```

Table 15–7 lists and describes the attributes associated with a `destype` element.

Table 15–7   Attributes of the destype element

Attribute	Valid values	Description
id	string	Required. A keyword, unique within a given distribution XML file, that identifies a particular file element. This can be a combination of a text string and one or more numbers, for example id="a1". The id value must always start with an alpha character.
name	string	Required. The name of the custom destination. For example, for a fax, this might be: name="fax" For a portal built with OracleAS Portal: name="oraclePortal"
instance	this|all	**Default:** all Used when the foreach element is also present. With a grouped report that is burst into separate reports, instance specifies whether the groups will be broken into separate destype instances according to each group instance (this) or all contained within the same destype instance (all). For example, if you custom destination type is a fax, instance="this" would mean a separate fax for each group instance, and instance="all" would mean one fax for all groups.

OracleAS Reports Services supports the creation and use of custom destination types (pluggable destinations) in the Reports Services environment. One way it does this is by allowing you to include calls to custom destinations in your distribution XML file. The distribution XML file provides a way to define custom destinations through property name/value pairs used in conjunction with the destype element.

15.3.10 property

Example

```
<foreach>
    <destype id="custom1" name="fax" instance="all">
```

```
        <include src="headerSection"/>
        <property name="number" value="914925551212"/>
    </destype>
</foreach>
```

Required/Optional

Optional. You can have as many properties as you require under a destype element.

Description

The property element allows for the inclusion of name/value pairs expressed in terms recognized by a custom destination type (destype). Properties are merely passed along to the destination handler. They serve no function within Reports Services. How you specify properties is entirely dependent on the requirements of your custom destination.

15.4 Distribution XML File Examples

This section provides examples, from simple to complex, of distribution XML elements. They are organized according to the main distribution.dtd elements:

- foreach Examples

- mail Examples

- file Examples

- printer Examples

15.4.1 foreach Examples

The examples in this section include:

- Single E-Mail with Report Groups as Separate Attachments

- Separate E-Mail for Each Group Instance

- Separate E-Mails with Separate Sections as Attachments

- Separate File for Each Section

- Separate Print Run for Each Report

15.4.1.1 Single E-Mail with Report Groups as Separate Attachments

In this example, each attachment contains the corresponding instance from the header, main, and trailer sections. That is, if the report is grouped on department_id, and the first department is department 10, the first attachment will be a report with header, main, and trailer sections all containing department 10 information. This example is valid only if the header, main, and trailer sections repeat on the same group instance, in this case department_id.

```
<mail id="a1" to="managers@mycompany.com" subject="New Hires">
    <foreach>
        <attach format="html" srcType="report" instance="this">
            <include src="report"/>
        </attach>
    </foreach>
</mail>
```

First of all, assume in this example that "managers@mycompany.com" goes to a mailing list that distributes to each department manager. If there are four departments: 10, 20, 30, and 40, the first attachment will contain header, main, and trailer sections corresponding to department 10; the second to 20; and so on. This example will yield one e-mail per recipient, each with four attachments.

15.4.1.2 Separate E-Mail for Each Group Instance

In this example, each recipient will receive a separate e-mail for each grouped report. For example, if the report is grouped on department_id, and there are four departments, one recipient will receive four e-mails, each with a separate department's report attached.

```
<foreach>
    <mail id="weeklies" to="managers@mycompany.com">
        <attach format="htmlcss" srcType="report" instance="this">
            <include src="mainSection"/>
        </attach>
    </mail>
</foreach>
```

15.4.1.3 Separate E-Mails with Separate Sections as Attachments

In this example, different sections repeat on different groups. The distribution is set up so that each recipient will receive a separate e-mail with attachment for each grouped main section and for each grouped trailer section.

```
<foreach>
    <mail id="a6" to="managers@mycompany.com" subject="Personnel Reports">
```

```
        <attach format="pdf" name="attach.pdf" srcType="report" instance="this">
           <include src="mainSection"/>
        </attach>
        <attach format="rtf" name="attach.rtf" srcType="report" instance="this">
           <include src="trailerSection"/>
        </attach>
    </mail>
</foreach>
```

15.4.1.4 Separate File for Each Section

In this example, a separate file is generated for each group instance. Groups repeat on department_id. Each file is named with the relevant department ID.

```
<foreach>
    <file id="a10" name="department_&amp;&lt;department_id&gt;.pdf"
    instance="this">
        <include src="mainSection"/>
    </file>
</foreach>
```

Assuming that there are four departments, 10 through 40, this example will result in the creation of four files, named in turn department_10.pdf, department_20.pdf, and so on.

The `format` attribute is not included in the `file` element because it is not required when the `srcType` is `file` or `text`. It is required when the `srcType` is `report`.

> **Note:** If you do not specify unique filenames through the use of variable values (see Section 15.2.2), in this example, each successively created file will overwrite the previously created file. That is, the department.pdf file for department 20 will overwrite the department.pdf file for department 10, and so on, until there is only one file left, department.pdf, with information from the last department report created (e.g., department 40).

15.4.1.5 Separate Print Run for Each Report

The way you specify a printer name differs between Windows and UNIX. The first example is for Windows. The second is for UNIX.

15.4.1.5.1 Windows In this example, assuming that the report is grouped on department_id, a report will be printed for each department.

```
<foreach>
    <printer id="a7" name="\\server_name\printer_name" instance="this">
        <include src="report"/>
    </printer>
</foreach>
```

15.4.1.5.2   UNIX In this example, assuming that the report is grouped on
department_id, a report will be printed for each department.

```
<foreach>
    <printer id="a7" name="printer_alias" instance="this">
        <include src="report"/>
    </printer>
</foreach>
```

15.4.2 mail Examples

The examples in this section include:

- E-Mail with a Whole Report as the Body

- E-Mail with a Section of a Report as the Body

- E-Mail with Two Report Sections as the Body

- E-Mail with External File as Body and Report as Attachment

- E-Mail with Whole Report and Grouped Sections Attached

- E-Mail to Relevant Manager and Department

15.4.2.1 E-Mail with a Whole Report as the Body

The report will comprise the content of this e-mail. That is, when recipients open
this e-mail, they will see the report.

```
<mail id="a5" to="managers@mycompany.com" subject="Quarterly Report">
    <body srcType="report" format="html">
        <include src="report"/>
    </body>
</mail>
```

15.4.2.2 E-Mail with a Section of a Report as the Body

A section of a report will comprise the content of this e-mail. That is, when
recipients open this e-mail, they will see a section of the report.

```
<mail id="a6" to="employees@mycompany.com">
```

```
 <body srcType="report" format="html">
    <include src="mainSection"/>
 </body>
</mail>
```

The `subject` attribute is not included in this `mail` element, so the default subject will be used: `Mail Sent From &Report`. At runtime, the variable `&Report` will be replaced with the name of the report.

15.4.2.3 E-Mail with Two Report Sections as the Body

Two sections of a report will comprise the body of this e-mail. That is, when recipients open this e-mail, they'll see two sections, headerSection and mainSection, joined together in one report.

```
<mail id="emp_addresses" to="employees@mycompany.com" subject="Employee Address
List">
    <body srcType="report" format="html">
       <include src="headerSection"/>
       <include src="mainSection"/>
    </body>
</mail>
```

15.4.2.4 E-Mail with External File as Body and Report as Attachment

The contents of the body for this email will be an external file, and the report will go along as an attachment. The path to the file is expressed differently for Windows and UNIX.

15.4.2.4.1 Windows

```
<mail id="XQRSN" to="accounting@mycompany.com" subject="Salaries"
    <body srcType="file">
       <include src="c:\mail\body.html"/>
    </body>
    <attach format="pdf" name="salaries.pdf" srcType="report">
       <include src="report"/>
    </attach>
</mail>
```

15.4.2.4.2 UNIX

```
<mail id="XQRSN" to="accounting@mycompany.com" subject="Salaries"
    <body srcType="file">
      <include src="/mail/body.html"/>
    </body>
```

```
        <attach format="pdf" name="salaries.pdf" srcType="report">
            <include src="report"/>
        </attach>
</mail>
```

15.4.2.5 E-Mail with Whole Report and Grouped Sections Attached

In this example, recipients receive one e-mail with multiple attachments: one attachment for each group instance and an additional attachment that contains the entire report. If the report is grouped on department_id and there are four departments, recipients will receive five attachments: one for each department and one whole report.

```
<mail id="grx90" to="sales@mycompany.com">
    <body srcType="text">
    Attached you will find the summary report and breakdown by department of
    weekly totals.
    </body>
    <attach format="rtf" name="myAttach.rtf" srcType="report">
        <include src="report"/>
    </attach>
    <foreach>
       <attach format="pdf" name="myattach.pdf" srcType="report"
       instance="this">
        <include src="mainSection"/>
        </attach>
    </foreach>
</mail>
```

15.4.2.6 E-Mail to Relevant Manager and Department

In this example, the manager for department 10 gets department 10's report; the manager for department 20 gets department 20's report; and so on. For this tag set to be valid, the variable must refer to a column that is included in the "repeat on" group used with the attached section. That is, if the section repeats on G_department_id, manager must be a column in that group.

```
<foreach>
    <mail id="mgr1090" to="&amp;&lt;manager&gt;@mycompany.com">
        <attach format="pdf" name="attach.pdf" srcType="report" instance="this">
            <include src="mainSection"/>
        </attach>
    </mail>
</foreach>
```

15.4.3 file Examples

Whenever you burst and distribute grouped reports to files, be sure to specify filenames with variable values based on the repeating group or some other variable information. Otherwise, you run the risk of having each successive file that is created overwrite the previously created file. For example, if you specify an output filename of department.pdf, and you output separate instances of each department's report, the second department.pdf file will overwrite the first department.pdf file; the third will overwrite the second; an so on. You will end up with only one report, that of the final department to be output. Instead, with grouped reports that you want to output separately according to each group instance, use variable values to specify filenames, for example: name="department_&<department_id>.pdf".

The examples in this section include:

- File for Whole Report

- File for Combined Report Sections

- File for Each Group of Combined Sections

- File for Each Report Group Instance

15.4.3.1 File for Whole Report

This example will yield one file named report.pdf that contains the entire report.

15.4.3.1.1 Windows

```
<file id="a1" name="c:\reports\report.pdf" format="pdf">
    <include src="report"/>
</file>
```

15.4.3.1.2 UNIX

```
<file id="a1" name="/reports/report.pdf" format="pdf">
    <include src="report"/>
</file>
```

15.4.3.2 File for Combined Report Sections

This example will yield one file named sections.pdf that contains a report consisting of the header section and the main section of the report.

```
<file id="a2" name="sections.pdf" format="pdf">
     <include="headerSection"/>
```

```
        <include="mainSection"/>
</file>
```

15.4.3.3 File for Each Group of Combined Sections

In this example, a separate file will be created for each repeating group. Each file will contain a report that combines the relevant group main and trailer sections. The main and trailer sections must repeat on the same group, and the variable file name must refer to a column contained within the "repeat on" group. That is, if the report repeats on department_id, and you have four departments, 10 through 40, then one file will contain the main and trailer sections of department 10; the next will contain the main and trailer sections of department 20; and so on. The variable value under name must refer to a column that is within the G_department_id group.

```
<foreach>
    <file id="file9" name="department_&amp;&lt;department_id&gt;.pdf"
    instance="this">
        <include src="mainSection"/>
        <include src="trailerSection"/>
    </file>
</foreach>
```

15.4.3.4 File for Each Report Group Instance

In this example, assuming the report is grouped on department_id and there are four departments, 10 through 40, you will end up with four files respectively named: department_10.pdf, department_20.pdf, department_30.pdf, and department_40.pdf.

```
<foreach>
    <file id="a20" name="department_&amp;&lt;department_id&gt;.pdf"
    instance="this">
        <include src="report"/>
    </file>
</foreach>
```

15.4.4 printer Examples

The examples in this section include:

- Print Whole Report

- Print Two Sections of a Report

- Print Grouped Report

- [Print Combined Sections for Each Group Instance](#)

- [Print Relevant Instance of a Report to Its Relevant Printer](#)

The way printer names are specified, differs between Windows and UNIX. Each example demonstrates both ways.

15.4.4.1 Print Whole Report

In this example, the entire report will be sent to the specified printer.

15.4.4.1.1 Windows

```
<printer id="a80" name="\\neptune\prtr20">
    <include src="report"/>
</printer>
```

15.4.4.1.2 UNIX

```
<printer id="a80" name="10th_floor_printer">
    <include src="report"/>
</printer>
```

15.4.4.2 Print Two Sections of a Report

In this example, two sections of a report will be sent to the printer.

15.4.4.2.1 Windows

```
<printer id="a1" name="\\neptune\prtr20">
    <include src="headerSection"/>
    <include src="mainSection"/>
</printer>
```

15.4.4.2.2 UNIX

```
<printer id="a1" name="10th_floor_printer">
    <include src="headerSection"/>
    <include src="mainSection"/>
</printer>
```

15.4.4.3 Print Grouped Report

In this example, one report will be printed. The report will be grouped by, for example, department_id. For this to work, all sections of the report must repeat on the same group.

15.4.4.3.1 Windows

```
<foreach>
    <printer id="prt20" name="\\neptune\prtr20" instance="all">
        <include src="report"/>
    </printer>
</foreach>
```

15.4.4.3.2 UNIX

```
<foreach>
    <printer id="prt20" name="10th_floor_printer" instance="all">
        <include src="report"/>
    </printer>
</foreach>
```

15.4.4.4 Print Combined Sections for Each Group Instance

This example will yield a number of print jobs: one for each group instance. The combined sections must repeat on the same group. If the report repeats on department_id, and you have four departments, 10 through 40, you will end up with four print jobs: one for department 10; one for department 20; and so on. The main and trailer sections must both repeat on department_id.

15.4.4.4.1 Windows

```
<foreach>
    <printer id="prt20" name="\\neptune\prtr20" instance="this">
        <include src="mainSection"/>
        <include src="trailerSection"/>
    </printer>
</foreach>
```

15.4.4.4.2 UNIX

```
<foreach>
     <printer id="prt20" name="10th_floor_printer" instance="this">
         <include src="mainSection"/>
         <include src="trailerSection"/>
     </printer>
</foreach>
```

15.4.4.5 Print Relevant Instance of a Report to Its Relevant Printer

For this example to work, the repeat on group must contain a column of printer names appropriate to the host platform (e.g., the printer_name column must

contain an appropriate printer alias on UNIX and a printer server/name combination on Windows). For example, if the report is grouped by department_id, then G_department_id must also have a printer_name column. Assuming the printer_names are tied to departments, then department 10's report would be printed on department 10's printer; department 20's report would be printed on department 20's printer; and so on.

```
<foreach>
    <printer id="a60" name="&amp;printer_name" instance="this">
       <include src="mainSection"/>
    </printer>
</foreach>
```

Each group instance equals a separate print job. Each print job goes to the relevant department's printer

15.5 Using a Distribution XML File at Runtime

The method for using a distribution XML file at runtime is essentially the same whether you use it in a URL or a command line. Use the commands:

```
distribute=yes destination=filename.xml
```

Where *filename* is the name of the distribution XML file. You are required to specify either the relative or absolute path of the XML file. For example, for Windows, you might specify:

```
distribute=yes destination=c:\ORACLE_HOME\reports\distribution\filename.xml
```

For UNIX, you might specify:

```
distribute=yes destination=ORACLE_HOME/reports/distribution/filename.xml
```

The paths in these examples are used for illustrative purposes only. There is no requirement for where you store your distribution XML files. You can store them wherever you like.

> **Note:** In some cases, Microsoft Internet Explorer ignores the mimetype of a URL's return stream and instead sets the type by looking at the URL. This can be a problem when you are using the distribution feature of OracleAS Reports Services because your URL might end with the destination parameter, for example:
>
> ```
> ...distribute=yes
> destination=c:\oracle\reports\distribution\mydist.
> xml
> ```
>
> In this scenario, your URL ends with the extension .xml and Internet Explorer treats the return stream as XML, when in fact it is HTML. As a result, you will receive a browser error. To work around this issue, you should never use recognized file extensions at the end of a URL. In the example above, you could switch the positions of the distribute and destination parameters in your URL.

For detailed information on running reports from command lines and URLs and using the cgicmd.dat file, see Chapter 13, "Running Report Requests".

15.6 Defining Custom/Pluggable Destinations

You can define a custom pluggable destination that can be used by Oracle Reports during distribution. The following examples illustrate the outlined destinations:

- Fax destination

- oraclePortal Destination

Fax destination

You can specify the destination as per the generic tag structure in the distribution XML file as follows:

Example 15–1   Generic Fax Destination

```
<destype id="faxdest" name="fax">
    <property name="number" value="123456789"/>
    <include src="report"/>
</destype>
```

oraclePortal Destination

You can specify the destination as per the generic tag structure in the distribution XML file as outlined in Example 15–2.

Example 15–2 illustrates a sample structure for DESTYPE=ORACLEPORTAL. When you push a report output to Portal using DESTYPE=ORACLEPORTAL, the report output will be created in the PAGEGROUP folder.

> **See Also:** Command Line Options
>
> For more information on the properties outlined in the examples.

Example 15–2   Simple Portal Destination

```
<destinations>
    <destype id="customforPortal" name="oraclePortal">
       <property name="outputpage" value="sample_report"/>
       <property name="statuspage" value="Reports_Status"/>
       <property name="desformat" value="pdf"/>
       <property name="pagegroup" value="REPORTS_OUTPUT"/>
       <property name="itemtitle" value="MyReport"/>
       <include src="report"/>
    </destype>
</destinations>
```

You can specify the custom destination in accordance with the generic destype tag structure for a pluggable destination. Alternatively, for ease of use, you can specify a custom, more specific tag structure.

> **Note:** Refer to destype=oraclePortal for more information on the limitation with using the oraclePortal destination.

Example 15–3   Custom Fax Destination

```
<fax id="faxdest" number="123456789">
      <include src="report"/>
</fax>
```

These tags are unknown to the distribution.dtd, therefore, they need to be mapped to the generic destype tag structure specified in the distribution.dtd.

Reports runtime cannot process the `<fax>` tag structure as illustrated here because the `<fax>` tag is not a standard destination specified in the `distribution.dtd` file. The custom tag structure must therefore be converted to the generic format as shown in Example 15–1.

Use the `distribution.xsl` file to transform user-defined custom tags in the distribution XML file to a format required by Reports runtime. Reports can understand only the generic `destype` tag structure for any pluggable destination.

The distribution.xsl file is an XML style sheet, located on both Windows and UNIX at *ORACLE_HOME*\reports\conf\distribution.xsl. You can modify this file by adding a template for translating your `destype` tag format to the required format defined in the `distribution.dtd` file.

To achieve this, you must specify a template for the fax destination in the `distribution.xsl` file. The template will be used to convert the `<fax>` tag structure to the generic `destype` structure. Your `distribution.xsl` entry might look like this:

```
<xsl:output doctype-system="distribution.dtd"/>

<xsl:template match = "/">
    <xsl:apply-templates match = "destinations" />
</xsl:template>

<xsl:template match="destinations">
    <destinations>

    <!--
    The Standard mail/file/printer/destype and foreach must be copied to the
    transformed xml. The foreach tag must be copied only if it is specified with
    file/mail/printer/destype tags.
    -->
        <xsl:copy-of select="mail"/>
        <xsl:copy-of select="file"/>
        <xsl:copy-of select="printer"/>
        <xsl:copy-of select="destype"/>
        <xsl:copy-of select="foreach"/>

        <!-- apply template for the sample FAX destination -->
        <xsl:apply-templates match = "fax" />

    </destinations>
</xsl:template>
```

```
<!--
    Sample Transformation Template for a FAX destination specified in the
    distribution.xml file

    <fax id="FAXDEST" number="123456789">
        <include src="report"/>
    </fax>
-->

<xsl:template match="fax">
    <!-- create a new destype element -->
    <xsl:element name="destype">
        <!--
        create an ID attribute and copy the value from the ID given for the fax
        destination
        -->
    <xsl:attribute name="id">
        <xsl:value-of select="@id"/>
    </xsl:attribute>

    <!-- create a Name attribute with a fax as it's value -->
    <xsl:attribute name="name">fax</xsl:attribute>
    <!--
    create a Property Attribute with name / value attribute pairs property tag
    is created for number attribute. Similarly create more property tags for any
    other attribute you add to the FAX destination
    -->
<xsl:element name="property">
        <xsl:attribute name="name">number</xsl:attribute>
        <xsl:attribute name="value">
            <xsl:value-of select="@number"/>
        </xsl:attribute>
</xsl:element>

        <!-- copy the include tag as it is -->
        <xsl:copy-of select="include"/>
    </xsl:element>
    <!-- end of template -->
</xsl:template>

</xsl:stylesheet>
```

> **Note:** All you need to do after you modify the XSL file is save it back to the same location under the same file name. Reports will automatically look for this XSL file when resolving distributions.

15.7 Limitations with using distribution

15.7.1 destype=oraclePortal

From Oracle9*i* Reports Release 2 (9.0.2) onwards, Reports Server supports OracleAS Portal as a destination. By using DESTYPE=ORACLEPORTAL, you can push a report to an output page specified in OracleAS Portal.

> **See Also:** Command Line Options and the *Reports Builder Online Help* for more information on the DESTYPE command line keyword.

However, there are a few limitations in using this destination:

- The ORACLEPORTAL destination cannot be used with distribution. Instead, you can use DESTYPE=WEBDAV for advanced XML based distribution to OracleAS Portal.

> **Note:** Ensure that the OracleAS Portal instance should be WebDAV-enabled. Refer to the *OracleAS Portal online help* for more information on how to enable WebDAV.
>
> For more information on how to use WebDAV for distribution to OracleAS Portal, refer to the Tech note at:
>
> `http://webiv.oraclecorp.com/cgi-bin/webiv/do.pl/Get?WwwID=note:241821.`

- The DESTYPE=ORACLEPORTAL destination cannot be used with the rwrun executable as it causes Reports Server to stop responding. Use this destination only with rwservlet, rwclient, or rwcgi.

15.7.2 XML and delimited outputs

Using reference parameters for report bursting (i.e., by specifying
`instance=this`) is not supported for `DESFORMAT=XML` and
`DESFORMAT=DELIMITED`. If used, it results in the following error message:

```
REP-34310 "Reference parameter not allowed in distribution list for XML
destination files"
```

However, these formats can be used in distribution without specifying a reference
parameter.

16

Customizing Reports with XML

Extensible Markup Language (XML) is designed to improve the functionality of the Web by providing a method to promote detailed information identification. It is actually a metalanguage (a language used for describing other languages) and can be used to design customized markup languages for different type of documents.

XML documents are composed of both markup and content:

- **Elements** are the building blocks of XML. An element instance is a structure that contains tags (a main tag and appropriate nested tags), attributes, and the element's content nested between the tags.

- **Tags** are used to define the element and the content within it.

- **Attributes** provide extra information for each tag.

XML customizations enable you to modify reports at runtime without changing the original report. With the addition of the CUSTOMIZE command to your runtime command line, you can call a customization file to add to or change a report's layout or data model. One XML customization file can perform all of these tasks or any combination of them. You can even use XML to build a report data model for inclusion in a custom JSP-based report.

By creating and applying different XML customizations, you can alter the report output on a per user or per user group basis. You can use the same report to generate different output depending upon the audience.

When you apply an XML customization to a report, you have the option of saving the combined definition to a file. As a result, you can use XML customizations to make batch updates to existing reports. You can quickly update a large number of reports without having to open each file in Reports Builder.

OracleAS Reports Services extends the possible types of Oracle Reports XML customizations by enabling you to create an entire reports data model in XML. This

includes the creation of multiple data sources, linking between data sources, and group hierarchies within each data source. Data model support via Oracle Reports XML customization means that any data model that can be created with Reports Builder can now be created by specifying XML. Additionally, all properties that can be set against data model objects can now be set using XML.

This chapter discusses the ways you can use XML to customize reports on the fly and to build data models. It includes the following sections:

- Customization Overview
- Creating XML Customizations
- Creating XML Data Models
- Using XML Files at Runtime
- Debugging XML Report Definitions

This chapter lists and provides examples of the supported elements in the `reports.dtd` file. However, only some of the attributes of these elements are listed.

For more information, either on the additional attributes or on the Oracle Reports XML elements, tags, and attributes, refer to the following sources:

- The `reports.dtd` file lists all the Oracle Reports XML elements, tags, and attributes and, where present, the attributes' default values. The `reports.dtd` file is located in *ORACLE_HOME*\reports\dtd\ on both Windows and UNIX platforms. Many of the sub-elements include symbols that denote usage rules. For example:

    - A plus sign (+) means you can have one or more of this type of element in your XML file.
    - An asterisk (*) means you can have from zero to many of this type of element in your XML file.
    - A question mark (?) means you can have either zero or one of this type of element in your XML file.
    - No mark means the element is required, and you can have one and only one of this type of element in your XML file.

    If multiple sub-elements are enclosed in parentheses and followed by a symbol, the symbol applies to all enclosed sub-elements.

- Topic "Oracle Reports XML tags" in the **Reference** section of the *Reports Builder online help* for descriptions of selected tags.

- Build a report that includes the type of customization you are trying to build, save the report as XML, and view the saved file in a text editor. This provides an excellent means of seeing Oracle Reports XML in action and provides you with examples of the more complex models you may wish to build.

16.1 Customization Overview

By using the Oracle Reports XML tags, you can customize reports created using Reports Builder.

> **Note:** Although it is possible to create an entire report manually using the Oracle Reports XML tags, only manually created customizations and data models are documented and supported.

Creating and applying an XML customization is a three-step process:

1. Create a customization file using Oracle Reports XML tags.

   You can create this customization by building a report using Reports Builder then saving your report as XML. You can also build the customization manually, with any sort of text editor or a sophisticated XML editor, as long as you include the XML tags that are required for the particular Oracle Reports customization.

   For descriptions of selected Oracle Reports XML tags, see opic "Oracle Reports XML tags" in the **Reference** section of the *Reports Builder online help* for descriptions of selected tags.

2. Store the XML customization in a location that is accessible to OracleAS Reports Services.

3. Apply the XML customization to another report with the CUSTOMIZE command line keyword or the SRW.APPLY_DEFINITION built-in procedure, or run the XML customization by itself (if it contains a complete report definition) with the REPORT (or MODULE) command line keyword.

   See Also: Command Line Options

   For information on using the command line keywords.

16.2 Creating XML Customizations

This section provides examples of various report customizations. It includes examples of:

- Required XML Tags
- Changing Styles
- Changing a Format Mask
- Adding Formatting Exceptions
- Adding Program Units and Hyperlinks
- Adding a New Query and Using the Result in a New Header Section
- Encoding the URL

16.2.1 Required XML Tags

Every XML customization must contain the following required tag pair:

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

For example, the following is the most minimal XML customization possible:

```
<report name="emp" DTDVersion="9.0.2.0.0">
</report>
```

This XML customization would have a null effect if applied to a report because it contains nothing. It can be parsed because it has the needed tags, but it is useful only as an example of the required tags.

The `<report>` tag indicates the beginning of the report customization, its name, and the version of the Data Type Dictionary (DTD) file that is being used with this XML customization. The `</report>` tag indicates the end of the report customization.

The `report` tag's `name` attribute can be any name you wish, either the name of the report the XML file will customize, or any other name.

This example represents a minimal use of the `<report>` tag. The `<report>` tag also has many attributes, most of which are implied and need not be specified. The only required `<report>` attribute is `DTDVersion`.

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>`

The `data` tag has no accompanying attributes. The `layout` tag has two attributes, both of which are required: `panelPrintOrder` and `direction`. If you use the default values for these attributes (respectively `acrossDown` and `default`), you don't need to specify them. Examples of the `data` and `layout` elements are provided in the following sections.

16.2.2 Changing Styles

The example in this section demonstrates the use of XML to change the fill and line colors used for report fields `F_Mincurrent_pricePersymbol` and `F_Maxcurrent_pricePersymbol`.

```
<report name="anyName" DTDVersion="9.0.2.0.0">
    <layout>
        <section name="main">
            <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>
</report>
```

We assume in this example that the `section` and `field` tags' name attributes match the names of fields in the Main section of the report this XML file will customize. In keeping with this assumption, the other attributes of the `field` tag will be applied only to the fields of the same name in the report's Main section.

16.2.3 Changing a Format Mask

The example in this section demonstrates the use of XML to change the format mask used for a report field `f_trade_date`.

```
<report name="anyName" DTDVersion="9.0.2.0.0">
    <layout>
        <section name="main"
            <field name="f_trade_date"
```

```
            source="trade_date"
            formatMask="MM/DD/RR"/>
        </section>
    </layout>
</report>
```

Notice that the `field` tag provides its own closure (`/>`). If the `field` tag used additional sub-tags, you would close it with `</field>`.

16.2.4 Adding Formatting Exceptions

The example in this section demonstrates the use of XML to add a formatting exception to highlight values greater than 10 in a report's `f_p_e` and `f_p_e1` fields.

```
<report name="anyName" DTDVersion="9.0.2.0.0">
    <layout>
        <section name="main">
            <field name="f_p_e" source="p_e">
                <exception textColor="red">
                    <condition source="p_e" operator="gt" operand1="10"/>
                </exception>
            </field>
            <field name="f_p_e1" source="p_e">
                <exception textColor="blue">
                    <condition source="p_e" operator="gt" operand1="10"/>
                </exception>
            </field>
        </section>
    </layout>
</report>
```

In this example, the value for `operator` is `gt`, for greater than. Operators include those listed in Table 16–1:

Table 16–1   Values for the operator attribute

Operator	Usage
eq	equal
lt	less than
lteq	less than or equal to
neq	not equal to
gt	greater than

Table 16–1   Values for the operator attribute

Operator	Usage
gteq	greater than or equal to
btw	between
notBtw	not between
like	like
notLike	not like
null	null
notNull	not null

Notice also that, unlike the previous example, the `field` tags in this example uses sub-tags, and, consequently, closes with `</field>`, rather than a self-contained closure (`/>`).

16.2.5 Adding Program Units and Hyperlinks

The example in this section demonstrates the use of XML to add a program unit to a report, which in turn adds a hyperlink from the employee social security number (:SSN) to employee details.

```
<report name="anyName" DTDVersion="9.0.2.0.0">
    <layout>
        <section name="header">
            <field name="F_ssn1" source="ssn1">
              <advancedLayout formatTrigger="F_ssn1FormatTrigger"/>
            </field>
        </section>
        <section name="main">
              <field name="F_ssn" source="ssn">
                 <advancedLayout formatTrigger="F_ssnFormatTrigger"/>
              </field>
        </section>
    </layout>
    <programUnits>
        <function name="F_ssn1FormatTrigger">
         <textSource>
         <![CDATA[
            function F_ssn1FormatTrigger return boolean is
                begin
```

```
                    SRW.SET_HYPERLINK('#EMP_DETAILS_&<' || LTRIM(TO_CHAR(:SSN))
                    || '>');
                    return (TRUE);
                end;
        ]]>
        </textSource>
      </function>
      <function name="F_ssnFormatTrigger">
       <textSource>
       <![CDATA[
          function F_ssnFormatTrigger return boolean is
             begin
                SRW.SET_LINKTAG('EMP_DETAILS_&<' || LTRIM(TO_CHAR(:SSN)) ||
                '>');
                return (TRUE);
             end;
        ]]>
        </textSource>
      </function>
    </programUnits>
</report>
```

A CDATA tag is used around the PL/SQL to distinguish it from the XML. Use the same tag sequence when you embed HTML in your XML file. In this example, the functions are referenced by name from the formatTrigger attribute of the advancedLayout tag.

16.2.6 Adding a New Query and Using the Result in a New Header Section

The example in this section demonstrates the use of XML to add a new query to a report and a new header section that makes use of the query result.

```
<report name="ref" DTDVersion="9.0.2.0.0">
    <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="BLAFbeige.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>
```

This example XML can be run by itself because it has both a data model and a complete layout.

Use aliases in your SELECT statements to ensure the uniqueness of your column names. If you do not use an alias, then the default name of the report column is used 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, which requires you to supply the correct name of the source column (the field).

The labelAttribute element defines the formatting for the field labels in the layout. Because it lies outside of the open and close field tag, it applies to all the labels in the tabular layout. If you wanted it to pertain to only one of the fields, then you place it inside the tag pair. If there is both a global and local labelAttribute element (one outside and one inside the tag pair), the local overrides the global.

16.2.7 Encoding the URL

To ensure that spaces and control characters are passed correctly, you may need to turn URL encoding on or off for the fields in your report. You can turn URL encoding on or off with the RW:FIELD tag in a report:

```
<rw:field
...
urlEncode=yes|no
...
/>
```

The default value for `urlEncode` is `no`.

16.3 Creating XML Data Models

OracleAS Reports Services introduces a greater level of sophistication in the types of data models you can create using Oracle Reports XML tags. Use XML for:

- Creating Multiple Data Sources
- Linking Between Data Sources
- Creating Group Hierarchies within Each Data Source
- Creating Cross-product (Matrix) Groups
- Creating Formulas, Summaries, and Placeholders at Any Level
- Creating Parameters

This section provides examples of these uses of XML.

In addition to these data model types, OracleAS Reports Services provides support for using PL/SQL in your XML. This includes support for local program units, report-level triggers, and attached PL/SQL libraries.

16.3.1 Creating Multiple Data Sources

The `<data>` tag now supports the creation of multiple data sources as well as the new pluggable data sources. Each data source is enclosed within its own `<dataSource>` tag. The data type definition for the `dataSource` element is:

```
<!ELEMENT dataSource
    ((select|plugin|plsql),
    comment?,
    displayInfo?,
    formula*,
    group*)>
<!ATTLIST dataSource
    name CDATA #IMPLIED
    defaultGroupName CDATA #IMPLIED
    maximumRowsToFetch CDATA #IMPLIED>
```

The following example creates two SQL data sources and names them *Q_1* and *Q_2*. It also creates all the necessary columns for the data sources and the default

group—giving the group the specified `defaultGroupName` or defaulting its own name if `defaultGroupName` is not specified.

```
<report name="anyname" DTDVersion="9.0.2.0.0">
    <data>
        <dataSource name="Q_1" defaultGroupName="G_DEPARTMENTS">
            <select>
                select * from departments
            </select>
        </dataSource>
        <dataSource name="Q_2" defaultGroupName="G_EMPLOYEES">
            <select>
                select * from employees
            </select>
        </dataSource>
    </data>
</report>
```

16.3.2 Linking Between Data Sources

In the presence of multiple data sources, it may be desirable to link the data sources together to create the appropriate data model. Oracle Reports data model link objects have also been exposed through Oracle Reports XML. They support both group- and column-level links. You can specify any number of links to create the required data model.

The data type definition for the `link` element is:

```
<!ELEMENT link EMPTY>
<!ATTLIST link
    name CDATA #IMPLIED
    parentGroup CDATA #IMPLIED
    parentColumn CDATA #IMPLIED
    childQuery CDATA #IMPLIED
    childColumn CDATA #IMPLIED
    condition (eq|lt|neq|gt|gteq|like|notLike) "eq"
    sqlClause (startWith|having|where) "where">
```

The `link` element is placed within a `data` element and can link any two `dataSource` objects defined within the `data` element. For example:

```
<report name="anyname" DTDVersion="9.0.2.0.0">
    <data>
        <dataSource name="Q_1" defaultGroupName="G_DEPARTMENTS">
            <select>
```

```
                    select * from departments
                </select>
            </dataSource>
            <dataSource name="Q_2" defaultGroupName="G_EMPLOYEES">
                <select>
                    select * from employees
                </select>
            </dataSource>
            <link name="L_1" parentGroup="G_DEPARTMENTS"
                parentColumn="DEPARTMENT_ID" childQuery="Q_2"
                childColumn="DEPARTMENT_ID1" condition="eq" sqlClause="where"/>
        </data>
</report>
```

Within the `link` element, Oracle Reports defaulting mechanism recognizes `DEPARTMENT_ID1` as an alias to the `DEPARTMENT_ID` column in the `EMPLOYEES` table without your having to explicitly create such an alias.

16.3.3 Creating Group Hierarchies within Each Data Source

With OracleAS Reports Services, the complete group hierarchy is available to you. You can specify all the columns within each group and break the order of those columns. You can use formulas, summaries, and placeholders to further customize the objects within groups.

The data type definition for the `group` element is:

```
<!ELEMENT group
    (field|exception|rowDelimiter|xmlSettings|displayInfo|dataItem|formula|
    summary|placeholder|filter|comment)*>
<!ATTLIST group
    name CDATA #IMPLIED
    fillColor CDATA #IMPLIED
    lineColor CDATA #IMPLIED
    formatTrigger CDATA #IMPLIED>
```

The following example demonstrates the use of a `group` element to create a break group under a data source.

```
<report name="anyname" DTDVersion="9.0.2.0.0">
    <data>
        <dataSource name="Q_1">
            <select>
                select * from employees
            </select>
```

```
                    <group name="G_DEPARTMENTS">
                        <dataItem name="DEPARTMENT_ID"/>
                    </group>
                    <group name="G_EMPLOYEES">
                        <dataItem="EMPLOYEE_ID"/>
                        <dataItem="FIRST_NAME"/>
                        <dataItem="LAST_NAME"/>
                        <dataItem="JOB_ID"/>
                        <dataItem="MANAGER_ID"/>
                        <dataItem="HIRE_DATE"/>
                        <dataItem="SALARY"/>
                        <dataItem="COMMISSION_PCT"/>
                    </group>
                </dataSource>
            </data>
        </report>
```

16.3.4 Creating Cross-product (Matrix) Groups

Cross-product groups allow you to define a matrix of any number of groups in the data model. The dimension groups in a cross product may exist in the same data source or may be combined from different data sources to create a matrix. In support of this flexibility, the `<crossProduct>` tag is placed within the `<data>` tag after all the data sources and groups have been created.

The data type definition for the `crossProduct` element is:

```
<!ELEMENT crossProduct
    (xmlSettings|displayInfo|dimension|(formula|summary|placeholder)*|comment)*>
<ATTLIST crossProduct
    name CDDATA #IMPLIED
    mailText CDDATA #IMPLIED>
```

The following example demonstrates the creation of a single-query matrix.

```
<report name="anyname" DTDVersion="9.0.2.0.0">
    <data>
        <dataSource name="Q_1">
            <select>
                select * from employees
            </select>
            <group name="G_DEPARTMENTS">
                <dataItem name="DEPARTMENT_ID"/>
            </group>
            <group name="G_JOB_ID>
```

```
                    <dataItem name="JOB_ID"/>
            </group>
            <group name="G_MANAGER_ID">
                <dataItem name="MANAGER_ID"
            </group>
            <group name="G_EMPLOYEE_ID">
                <dataItem name="EMPLOYEE_ID"/>
                <dataItem name="FIRST_NAME"/>
                <dataItem name="LAST_NAME"/>
                <dataItem name="HIRE_DATE"/>
                <dataItem name="SALARY"/>
                <dataItem name="COMMISSION_PCT"/>
            </group>
        </dataSource>
        <crossProduct name="G_Matrix">
            <dimension>
                <group name="G_DEPARTMENTS">
            </dimension>
            <dimension>
                <group name="G_JOB_ID">
            </dimension>
            <dimension>
                <group name="G_MANAGER_ID">
            </dimension>
        </crossProduct>
    </data>
</report>
```

16.3.5 Creating Formulas, Summaries, and Placeholders at Any Level

You can place formulas, summaries, and placeholders at any level within the data model. Additionally, you have complete control over all the attributes for each of these objects.

The following example demonstrates the creation of a report-level summary whose source is based on a group-level formula column.

```
<report name="anyname" DTDVersion="9.0.2.0.0">
    <data>
        <dataSource name="Q_1">
            <select>
                select * from employees
            </select>
            <group name="G_EMPLOYEES">
                <dataItem="EMPLOYEE_ID"/>
```

```
                        <dataItem name="EMPLOYEE_ID"/>
                        <dataItem name="FIRST_NAME"/>
                        <dataItem name="LAST_NAME"/>
                        <dataItem name="HIRE_DATE"/>
                        <dataItem name="SALARY"/>
                        <dataItem name="COMMISSION_PCT"/>
                        <dataItem name="DEPARTMENT_ID"/>
                        <formula name="CF_REMUNERATION" source="cf_1formula"
                            datatype="number" width="20" precision="10"/>
                </group>
            </dataSource>
            <summary name="CS_REPORT_LEVEL_SUMMARY" function="sum" width="20"
                precision="10" reset="report" compute="report"/>
        </data>
        <programUnits>
            <function name="cf_1formula" returnType="number">
                <textSource>
<![CDATA[
function CF_1Formula return Number is
begin
    return (:salary + nvl(:commission_pct,0));
end;
]]>
                </textSource>
            </function>
        </programUnits>
</report>
```

16.3.6 Creating Parameters

In Oracle Reports XML, the `parameter` element is placed between open and close `data` tags. The data type definition for the `parameter` element is:

```
<!ELEMENT parameter (comment?|listOfValues?)>
<!ATTLIST parameter
    name CDATA #REQUIRED
    datatype (number|character|date) "number"
    width CDATA "20"
    scale CDATA "0"
    precision CDATA "0"
    initialValue CDATA #IMPLIED
    inputMask CDATA #IMPLIED
    validationTrigger CDATA #IMPLIED
    label CDATA #IMPLIED
    defaultWidth CDATA #IMPLIED
```

```
                    defaultHeight CDATA #IMPLIED>
```

The following example demonstrates a dynamic list of values (LOV), an initial
value, and a validation trigger.

```
<report name="anyname" DTDVersion="9.0.2.0.0">
    <data>
        <dataSource name="Q_1" defaultGroupName="G_DEPARTMENTS">
            <select>
                select * from departments
            </select>
        </dataSource>
        <parameter name="P_LAST_NAME" datatype="character" precision="10"
            initialValue="SMITH" validationTrigger="p_last_namevalidtrigger"
            defaultWidth="0" defaultHeight="0">
            <listOfValues restrictToList="yes">
                <selectStatement hideFirstColumn="yes">
                <![CDATA[select last_name, 'last_name||'-'||employee_id'
                    from employees]]>
                </selectStatement>
            </listOfValues>
        </parameter>
    </data>
    <programUnits>
        <function name="p_last_namevalidtrigger" returnType="character">
            <textSource>
<![CDATA[function P_LAST_NAMEValidTrigger return boolean is last_name
char(20);
begin
    select count(*) into last_name from employees
    where upper(last_name)=upper(:p_last_name);
        exception when OTHERS then return(FALSE);
        end;
    return(TRUE);
end;
]]>
            </textSource>
        </function>
    </programUnits>
</report>
```

16.4 Using XML Files at Runtime

Once you have created your Oracle Reports XML customization file, you can use it in the following ways:

- You can apply XML report definitions to RDF or other XML files at runtime by specifying the CUSTOMIZE command line keyword or the SRW.APPLY_ DEFINITION built-in procedure. Refer to Applying an XML Report Definition at Runtime for more information.

- You can run an XML report definition by itself (without another report) by specifying the REPORT (or MODULE) command line keyword. Refer to Running an XML Report Definition by Itself for more information.

- You can use RWCONVERTER to make batch modifications using the CUSTOMIZE command line keyword. Refer to Performing Batch Modifications for more information.

The following sections describe each of the cases in more detail and provide examples.

16.4.1 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 keyword or the SRW.APPLY_DEFINITION built-in procedure. CUSTOMIZE can be used with rwclient, rwrun, rwbuilder, rwconverter, and URL report requests.

> **Note:** Refer to Performing Batch Modifications for more information about using CUSTOMIZE with rwconverter.

16.4.1.1 Applying One XML Report Definition

The following command line sends a job request to OracleAS Reports Services and applies an XML report definition, EMP.XML, to an RDF file, emp.rdf. In this example, the CUSTOMIZE command refers to a file located in a Windows directory path. For UNIX, specify the path according to UNIX standards (i.e., myreports/emp.xml).

```
rwclient report=emp.rdf customize=\myreports\emp.xml
    USERID=username/password@my_db destype=file desname=emp.pdf
    desformat=PDF server=server_name
```

When you use RWRUN, the Reports Runtime command, the equivalent command line would be:

```
RWRUN USERID=username/password@my_db report=emp.rdf
   customize=\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 options to create a trace file. For example:

```
TRACEFILE=emp.log TRACEMODE=trace_replace TRACEOPT=trace_app
```

> **Note:** Unless you care to change the default, it isn't necessary to include a trace in the command line if you have specified a default trace option in the Reports Server configuration file.

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

16.4.1.2 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 keyword. The following command line sends a job request to OracleAS Reports Services that applies two XML report definitions, EMP0.XML and EMP1.XML, to an RDF file, EMP.RDF:

```
rwclient report=emp.rdf
   customize="(d:\corp\myreports\emp0.xml,d:\corp\myreports\emp1.xml)"
   userid=username/password@my_db destype=file desname=emp.pdf
   desformat=PDF server=server_name
```

> **Note:** In this example, the CUSTOMIZE command entry demonstrates a directory path to files stored on a Windows platform. For UNIX, use that platform's standard for specifying directory paths (i.e., forward slashes instead of backward).

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

```
rwrun report=emp.rdf
   customize="(D:\CORP\MYREPOORTS\EMP0.XML,D:\CORP\MYREPORTS\EMP1.XML)"
   userid=username/password@my_db destype=file desname=emp.pdf
   desformat=PDF
```

16.4.1.3 Applying an XML Report Definition in PL/SQL

To apply an XML report definition to an RDF file in PL/SQL, use the SRW.APPLY_
DEFINITION and SRW.ADD_DEFINITION built-ins in the BeforeForm or
AfterForm trigger. The following sections provide examples of these built-ins.

16.4.1.3.1 Applying an XML Definition Stored in a File To apply XML that is stored in the
file system to a report, use the SRW.APPLY_DEFINITION built-in procedure in the
BeforeForm or AfterForm triggers of the report.

On **Windows**:

```
SRW.APPLY_DEFINITION ('\ORACLE_HOME\TOOLS\DOC\US\RBBR\COND.XML');
```

On **UNIX**:

```
SRW.APPLY_DEFINITION ('ORACLE_HOME/TOOLS/DOC/US/RBBR/COND.XML');
```

When the report is run, the trigger executes and the specified XML file is applied to
the report.

16.4.1.3.2 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 the SRW.APPLY_DEFINITION
built-in procedure.

The following example illustrates how to build up and apply several definitions in
memory based upon parameter values entered by the user. The PL/SQL in this
example is used in the AfterParameterForm trigger of a report called videosales_
custom.rdf.

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.

The following tips are useful when looking at this example:

- Each time you use the SRW.APPLY_DEFINITION built-in procedure, 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 (4000 bytes), 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="9.0.2.0.0">');
IF :MONEY_FORMAT='$NNNN.00' THEN SRW.ADD_DEFINITION('<layout>');
    SRW.ADD_DEFINITION('<section name="main">');
    SRW.ADD_DEFINITION('<field name="F_TOTAL_PROFIT" source="TOTAL_PROFIT"
    formatMask="LNNNNNNNNNNN0D00"/>');
    SRW.ADD_DEFINITION('<field name="F_TOTAL_SALES" source="TOTAL_SALES"
    formatMask="LNNNNNNNNNNN0D00"/>');
    SRW.ADD_DEFINITION('<field name="F_TOTAL_COST" source="TOTAL_COST"
    formatMask="LNNNNNNNNNNN0D00"/>');
    SRW.ADD_DEFINITION('<field name="F_SumTOTAL_PROFITPerCITY" source="SumTOTAL_
    PROFITPerCITY"
        formatMask="LNNNNNNNNNNN0D00"/>');
    SRW.ADD_DEFINITION('<field name="F_SumTOTAL_SALESPerCITY" source="SumTOTAL_
    SALESPerCITY"
        formatMask="LNNNNNNNNNNN0D00"/>');
    SRW.ADD_DEFINITION('<field name="F_SumTOTAL_COSTPerCITY" source="SumTOTAL_
    COSTPerCITY"
        formatMask="LNNNNNNNNNNN0D00"/>');
    SRW.ADD_DEFINITION('</section>');
    SRW.ADD_DEFINITION('</layout>');
ELSIF :MONEY_FORMAT='$NNNN' THEN SRW.ADD_DEFINITION('<layout>');
    SRW.ADD_DEFINITION('<section name="main">');
    SRW.ADD_DEFINITION('<field name="F_TOTAL_PROFIT" source="TOTAL_PROFIT"
    formatMask="LNNNNNNNNNNN0"/>');
    SRW.ADD_DEFINITION('<field name="F_TOTAL_SALES" source="TOTAL_SALES"
    formatMask="LNNNNNNNNNNN0"/>');
    SRW.ADD_DEFINITION('<field name="F_TOTAL_COST" source="TOTAL_COST"
    formatMask="LNNNNNNNNNNN0"/>');
    SRW.ADD_DEFINITION('<field name="F_SumTOTAL_PROFITPerCITY" source="SumTOTAL_
    PROFITPerCITY" formatMask="LNNNNNNNNNNN0"/>');
    SRW.ADD_DEFINITION('<field name="F_SumTOTAL_SALESPerCITY" source="SumTOTAL_
```

```
        SALESPerCITY" formatMask="LNNNNNNNNNNNN0"/>');
        SRW.ADD_DEFINITION('<field name="F_SumTOTAL_COSTPerCITY" source="SumTOTAL_
        COSTPerCITY" formatMask="LNNNNNNNNNNNN0"/>');
        SRW.ADD_DEFINITION('</section>');
        SRW.ADD_DEFINITION('</layout>');
END IF;
SRW.ADD_DEFINITION('</report>');
SRW.APPLY_DEFINITION;
SRW.ADD_DEFINITION('<report name="vidsales_hilite_costs" author="Generated"
DTDVersion="9.0.2.0.0">');
IF :HILITE_COSTS <> 'None' THEN SRW.ADD_DEFINITION('<layout>');
        SRW.ADD_DEFINITION('<section name="main">');
        SRW.ADD_DEFINITION('<field name="F_TOTAL_COST" source="TOTAL_COST">');
        SRW.ADD_DEFINITION('<exception textColor="red">');
        SRW.ADD_DEFINITION('<condition source="TOTAL_COST" operator="gt"
        operand1=":hilite_costs"/>');
        SRW.ADD_DEFINITION('</exception>');
        SRW.ADD_DEFINITION('</field>');
        SRW.ADD_DEFINITION('</section>');
        SRW.ADD_DEFINITION('</layout>');
END IF;
SRW.ADD_DEFINITION('</report>');
SRW.APPLY_DEFINITION;
SRW.ADD_DEFINITION('<report name="vidsales_hilite_sales" author="Generated"
DTDVersion="9.0.2.0.0">');
IF :HILITE_SALES <> 'None' THEN SRW.ADD_DEFINITION('<layout>');
        SRW.ADD_DEFINITION('<section name="main">');
        SRW.ADD_DEFINITION('<field name="F_TOTAL_SALES" source="TOTAL_SALES">');
        SRW.ADD_DEFINITION('<exception textColor="red">');
        SRW.ADD_DEFINITION('<condition source="TOTAL_SALES" operator="gt"
        operand1=":hilite_sales"/>');
        SRW.ADD_DEFINITION('</exception>');
        SRW.ADD_DEFINITION('</field>');
        SRW.ADD_DEFINITION('</section>');
        SRW.ADD_DEFINITION('</layout>');
END IF;
SRW.ADD_DEFINITION('</report>');
SRW.APPLY_DEFINITION;
SRW.ADD_DEFINITION('<report name="vidsales_hilite_profits" author="Generated"
DTDVersion="9.0.2.0.0">');
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;
```

16.4.2 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 (or MODULE) option. The following command line sends a job request to OracleAS Reports Services to run a report, emp.xml, by itself:

```
rwclient userid=username/password@my_db
    report=c:\corp\myreports\emp.xml
    destype=file desname=emp.pdf desformat=pdf
    server=server_name
```

When you use rwrun, the Reports Runtime command, the equivalent command line would be:

```
rwrun userid=username/password@my_db
  report=c:\corp\myreports\emp.xml
  destype=file desname=emp.pdf desformat=PDF
```

When you run an XML report definition in this way, you must specify an XML file extension. You could also apply an XML customization file to this report using the CUSTOMIZE option.

16.4.3 Performing Batch Modifications

If you have a large number of reports that need to be updated, then you can use the CUSTOMIZE command line keyword with RWCONVERTER to perform modifications in batch. Batch modifications are particularly useful when you must 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 Reports Builder, you can run RWCONVERTER 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, INVEN1.RDF, INVEN2.RDF, and MANU.RDF, and saves the revised definitions to new files, INVEN1_NEW.RDF, INVEN2_NEW.RDF, and MANU_NEW.RDF.

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

> **Note:** In this example, the CUSTOMIZE command entry demonstrates a directory path to files stored on a Windows platform. For UNIX, use that platform's standard for specifying directory paths (i.e., forward slashes instead of backward).

16.5 Debugging XML Report Definitions

The following features are available to help you debug your XML report files:

- XML Parser Error Messages
- Tracing Options
- rwbuilder
- Writing XML to a File for Debugging

These features are discussed in the following sections.

16.5.1 XML Parser Error Messages

The XML parser is part of Oracle's XML Development Kit (XDK), which is delivered with the core Oracle Database release. The XML parser is a Java package that checks the validity of XML syntax. The JAR files that contain the XML parser are automatically set up on install and are available to Oracle Reports.

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.

For more information on the XML parser, see the Oracle Technology Network, (`http://otn.oracle.com`). Search for *XML parser* or *XDK*. Information is also available in the documentation that came with your Oracle Database.

16.5.2 Tracing Options

When testing your XML report definition, it can be useful to run your report along with additional options to create a trace file. For example:

```
RWRUN <username>/<password>@<my_db>
    REPORT=\CORP\MYREPORTS\EMP.XML
    TRACEFILE=emp.log
    TRACEMODE=trace_replace
    TRACEOPT=trace_app
```

The last three options in this command line generate a trace file that provides a detailed listing of report processing. The default location for trace file logs is the same on Windows and UNIX platforms:

```
ORACLE_HOME\reports\logs\
```

> **Note:** In this example, the REPORT command entry and the path to the trace log demonstrate directory paths to files stored on a Windows platform. For UNIX, use that platform's standard for specifying directory paths,i.e., forward slashes instead of backward.

16.5.3 rwbuilder

When designing an XML report definition, it is sometimes useful to open it in Reports Builder. In Reports 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 Reports 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 Reports Builder, use the REPORT (or MODULE) keyword. For example:

```
rwbuilder userid=username/password@my_db report=c:\corp\myreports\emp.xml
```

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

```
rwbuilder userid=username/password@my_db report=emp.rdf
```

```
customize=c:\myreports\emp.xml
```

> **Note:** In this example, the REPORT command entry demonstrates a directory path to files stored on a Windows platform. For UNIX, use that platform's standard for specifying directory paths,i.e., forward slashes instead of backward.

In both cases, Reports Builder is opened with the XML report definition in effect. You can then use the various views of Reports Builder to determine if the report is being created or modified as you expected.

16.5.4 Writing XML to a File for Debugging

If you are using SRW.ADD_DEFINTION to build an XML report definition in memory, then it can be helpful to write the XML to a file for debugging purposes. The following example demonstrates a procedure that writes each line that you pass to it to the document buffer in memory and, optionally, to a file that you specify.

```
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="9.0.2.0.0">',
    custom_summary);
```

17

Using Event-Driven Publishing

Modern business processes often require the blending of automation into the work environment through the invocation of behind-the-scenes functions and procedures. Behind-the-scenes tasks can include the automatic production of output such as an invoice that prints automatically when an order is processed, a Web site that is automatically updated with current data, or an automatic e-mail with fresh report output when a transaction is completed.

Automatic output in response to events used to be a fairly complicated effort, particularly if you wished to produce the same results possible through interactive, RAD development tools, such as Oracle Reports Developer.

To address the requirement of automatic output, Oracle introduced a scheduling mechanism in OracleAS Reports Services that enabled the invocation of reports on a scheduled basis without requiring additional user interaction. But this left one requirement unresolved: the ability to automatically run a report in response to an event in the database, such as the insertion of a record or the change of a value.

With the OracleAS Reports Services Event-Driven Publishing API, you can automatically run a report in response to an event in the database, such as the insertion of a record or the change of a value. The Event-Driven Publishing API is a PL/SQL API that allows for the automatic submission of jobs to OracleAS Reports Services from within the database.

This chapter provides a look at the Event-Driven Publishing API and includes examples of its use. It includes the following sections:

- The Event-Driven Publishing API
- Debugging Applications that Use the Event-Driven Publishing API
- Invoking a Report From a Database Event
- Integrating with Oracle Advanced Queuing

17.1 The Event-Driven Publishing API

The Event-Driven Publishing API is a PL/SQL package that provides the basic functions required for the development of procedures that respond to events in the database. Event-driven jobs are submitted using the HTTP protocol. The server assigns a unique `job_ident` record to every call, useful for tracking the status of the job.

17.1.1 Elements of the API

The API consists of several key elements:

- The **SRW-Package** contains all relevant functions and procedures for submitting jobs, checking job status, and cancelling jobs, as well as manipulating parameter lists.

- The **ParamList-Type** defines a parameter list. A parameter list is the main vehicle for passing values when submitting a job. A parameter list is required for each job submission. It must contain several key parameters.

- The **ParamList-Object** is required for such features as Advanced Queuing, where a parameter list must be stored in the database so that it may be passed along with a message.

These API elements are discussed in more detail in the following sections.

The API is installed together with OracleAS Reports Services Security and OracleAS Portal, but neither is required. Installation scripts are also available separately should you want to install the API into a database that does not also hold Oracle Portal:

- `srwAPIins.sql` installs the Events-Driven Publishing API.

- `srwAPIgrant.sql` grants access privileges to the API. Run this script for each user to whom you will grant access to the API. If everyone may have access, you can run this once and grant access to PUBLIC.

- `srwAPIdrop.sql` removes the API.

17.1.2 Creating and Manipulating a Parameter List

A parameter list is a PL/SQL variable of type `SRW_PARAMLIST`. A variable of this type is an array of 255 elements of type `SRW_PARAMETER`, which itself consists of two attributes: NAME and VALUE. The API provides procedures for manipulating parameter lists, including:

- Add_Parameter

- Remove_Parameter

- Clear_Parameter_List

These procedures allow you to manipulate your parameter lists. They are discussed briefly in this section. You'll find more information in the Oracle Reports API documentation.

> **Note:** Look for upcoming information about Reports APIs on the Oracle Technology Network, (`http://otn.oracle.com`).

17.1.2.1 Add_Parameter

Whenever you use a parameter list for the first time, it must be initialized before you can add parameters to it. For example:

```
DECLARE
    myPlist SRW_PARAMLIST;
BEGIN
    myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
    srw.add_parameter(myPlist,'myParameter','myValue');
END;
```

Both attributes of a parameter (NAME and VALUE) are of type VARCHAR2 and may not exceed a length of 80 characters for the NAME and 255 characters for the value.

The ADD_PARAMETER function has a third—optional—attribute, called MODE. MODE determines whether a parameter will be overwritten or an error raised in the event that a parameter with the same name already exists. To specify that an error will be raised in the event of duplicate names, use the constant CHECK_FOR_EXISTANCE. This is the default value for the MODE attribute. To specify that a parameter will be overwritten in the event of duplicate names, use the constant OVERWRITE_IF_ EXISTS.

17.1.2.2 Remove_Parameter

Use REMOVE_PARAMETER to remove a parameter from a parameter list. Call the procedure, and pass the parameter list from which you want to remove a parameter along with the name of the parameter you want to remove.

For example:

```
DECLARE
    myPlist SRW_PARAMLIST;
BEGIN
    myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
    srw.add_parameter(myPlist,'myParameter','myValue');
    srw.remove_parameter(myPlist,'myParameter');
END;
```

17.1.2.3 Clear_Parameter_List

To remove ALL parameters from your list, use CLEAR_PARAMETER_LIST. For example:

```
DECLARE
    myPlist SRW_PARAMLIST;
BEGIN
    myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
    srw.add_parameter(myPlist,'myParameter','myValue');
    srw.clear_parameter_list(myPlist);
END;
```

This will remove all parameters from your list.

17.1.3 How to Submit a Job

A parameter list contains all vital parameters for submitting a job. The job type determines which parameters are required on the list to enable the Reports Server to process the request.

The listed parameters are the same ones that you must specify when you submit a job from a browser to the Reports Servlet. In such a case, if the job is a report you will need at least the following parameters but may have more:

- GATEWAY provides the URL to the Reports Servlet you will use to process the request.

- SERVER identifies the Reports Server to be used in conjunction with the servlet.

- REPORT identifies the report file to be run.

- USERID identifies the name and user ID of the person running the report.

- AUTHID provides authorization information in the event you are running against a secured server.

Each request returns a job_ident record that holds the information required to identify the job uniquely. This information is stored in variable of type SRW.JOB_IDENT. Be aware that this is a PACKAGE-TYPE and must be referenced SRW.JOB_IDENT; while the parameter list is an OBJECT-TYPE and must be referenced SRW_PARAMLIST.

For example:

```
DECLARE
   myPlist SRW_PARAMLIST;
   myIdent SRW.Job_Ident;
BEGIN
   myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
   srw.add_parameter(myPlist,'GATEWAY','http://…');
   srw.add_parameter(myPlist,'SERVER','mySVR');
   srw.add_parameter(myPlist,'REPORT','myReport.RDF');
   srw.add_parameter(myPlist,'USERID','me/secret');
   myIdent := srw.run_report(myPlist);
END;
```

The API method RUN_REPORT takes a parameter list that contains all vital information as input (via ADD_PARAMETER), creates and submits the request, and returns the job_ident record.

The job_ident record contains the following parameters:

- MyIdent.GatewayURL

- MyIdent.ServerName

- MyIdent.JobID

- MyIdent.AuthID

These parameters are needed by the SRW.REPORT_STATUS function to get status information for a submitted job.

17.1.4 How to Check for Status

The Event-Driven Publishing API provides a two-way communication with the Reports Server. You submit a job to the server, and you can query the status of this job from the server using the SRW.REPORT_STATUS function.

This function will return a record of type SRW.STATUS_RECORD that holds the same information you would see in the job-status display if you were using the Reports Servlet's SHOWJOBS command.

For example:

```
DECLARE
    myPlist SRW_PARAMLIST;
    myIdent SRW.Job_Ident;
    myStatus SRW.Status_Record;
BEGIN
    myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
    srw.add_parameter(myPlist,'GATEWAY','http://…');
    srw.add_parameter(myPlist,'SERVER','mySVR');
    srw.add_parameter(myPlist,'REPORT','MyReport.RDF');
    srw.add_parameter(myPlist,'USERID','me/secret');
    myIdent := srw.run_report(myPlist);
    myStatus := srw.report_status(myIdent);
END;
```

You can use the returned status record for fetching information about the status of your job.

17.1.5 Using the Servers' Status Record

The status record contains processing information about your job. It contains the same information found in the server queue (SHOWJOBS). Additionally, it contains information about the files produced for finished jobs and the lineage for scheduled jobs.

The most important information in the status record is the current job status and the status text, used in turn to check for runtime errors and their causes.

You can use timing information to determine if a job is subject to cancellation because it has exceeded its predicted time for completion.

One way to use the status record is to cancel a job. The Event-Driven Publishing API offers a method for cancelling a job that has been submitted to the server. This might be handy if you want to remove a job that has exceeded its allowed time to run or if you simply have scheduled jobs you want to cancel.

To cancel a job, use the following procedure:

```
DECLARE
    myPlist SRW_PARAMLIST;
    myIdent SRW.JOB_IDENT;
    myStatus SRW.STATUS_RECORD;
BEGIN
    myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
    SRW.ADD_PARAMETER(myPlist,'GATEWAY','http://…');
```

```
        SRW.ADD_PARAMETER(myPlist,'SERVER','mySVR');
        SRW.ADD_PARAMETER(myPlist,'REPORT','myReport.RDF');
        SRW.ADD_PARAMETER(myPlist,'USERID','me/secret');
        myIdent := SRW.RUN_REPORT(myPlist);
        myStatus := SRW.REPORT_STATUS(myIdent);
        if myStatus.StatusCode != srw.RUNNING then
        SRW.CANCEL_REPORT(myIdent);
END;
```

As evident in this example, you cancel a report by calling the CANCEL_REPORT procedure (SRW.CANCEL_REPORT) and passing it the job_ident record of the job you want to cancel. The procedure takes an optional parameter list to enable you to pass any additional parameters you might need.

17.2 Debugging Applications that Use the Event-Driven Publishing API

Because these processes all run behind the scenes, there is no actual place where debugging information is produced during normal execution. Therefore, the API has two procedures that toggle a special debugging mode that produces extensive debugging information via DBMS_OUTPUT:

- SRW.START_DEBUGGING

- SRW.STOP_DEBUGGING

To switch on debugging mode simply call SRW.START_DEBUGGING and to stop it call SRW.STOP_DEBUGGING. The debugging-mode must be started immediately before you run your actual logic. It stays on as long as the current instance of the package is loaded.

One way you can display this information is by setting SERVEROUT to ON in SQL*PLUS before you run your script.

In addition to this method of debugging, the API has a set of pre-defined exceptions to be used for error handling. You'll find examples of these exceptions in the srw_test.sql script provided with your OracleAS Reports Services installation. Additionally, see the Oracle Reports API reference documentation for a detailed explanation of these exceptions.

> **Note:**   Look for upcoming information about Reports APIs on the Oracle Technology Network, (http://otn.oracle.com).

17.3 Invoking a Report From a Database Event

Database triggers are the primary mechanism for invoking reports using the Event-Driven Publishing API. The Oracle database allows you to define various scopes of triggers that fire in response to various events. To submit a database-driven job, you use the code described in the previous sections within a database trigger.

There are many ways to use event-driven publishing. One way is to create security protocols using a trigger that fires whenever a grant is done or a user logs on or off. Another way is to create automated processes that respond to certain types of changes to data in a table. For example, a database trigger could fire when the status of an expense report changes to DONE; in turn, a report could automatically be sent to an employee's manager.

For example:

```
CREATE TRIGGER EXP_REP_TRG
    AFTER INSERT OR UPDATE on EXP_REP FOR EACH ROW
    myPlist SRW_PARAMLIST;
    myIdent SRW.JOB_IDENT;
BEGIN
    IF (:new.ExpStat = 'DONE') THEN
        myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
        SRW.ADD_PARAMETER(myPlist,'GATEWAY','http://…');
        SRW.ADD_PARAMETER(myPlist,'SERVER','fooSVR');
        SRW.ADD_PARAMETER(myPlist,'REPORT','foo.RDF');
        SRW.ADD_PARAMETER(myPlist,'USERID','foo/bar');
        SRW.ADD_PARAMETER(myPlist,'ExpenseID',:new.ExpID);
        myIdent := SRW.RUN_REPORT(myPlist);
    END IF;
END;
```

This trigger will fire after each update on the EXP_REP table. In the event the status changes to DONE, the report request is run.

If you want your request to run against a key specified in the cgicmd.dat file, specify the CMDKEY parameter in lieu of the REPORT parameter. If the key contains user ID information, you can omit the USERID parameter as well. For example:

```
CREATE TRIGGER EXP_REP_TRG
    AFTER INSERT OR UPDATE on EXP_REP FOR EACH ROW
    myPlist SRW_PARAMLIST;
    myIdent SRW.JOB_IDENT;
BEGIN
    IF (:new.ExpStat = 'DONE') THEN
```

```
        myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
        SRW.ADD_PARAMETER(myPlist,'GATEWAY','http://…');
        SRW.ADD_PARAMETER(myPlist,'SERVER','fooSVR');
        SRW.ADD_PARAMETER(myPlist,'CMDKEY','keyvalue');
        SRW.ADD_PARAMETER(myPlist,'ExpenseID',:new.ExpID);
        myIdent := SRW.RUN_REPORT(myPlist);
    END IF;
END;
```

Additionally, if you have defined an advanced distribution model via a distribution XML file, you can specify that file with the DESTINATION parameter. For example:

```
CREATE TRIGGER EXP_REP_TRG
    AFTER INSERT OR UPDATE on EXP_REP FOR EACH ROW
    myPlist SRW_PARAMLIST;
    myIdent SRW.JOB_IDENT;
BEGIN
    IF (:new.ExpStat = 'DONE') THEN
        myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
        SRW.ADD_PARAMETER(myPlist,'GATEWAY','http://…');
        SRW.ADD_PARAMETER(myPlist,'SERVER','fooSVR');
        SRW.ADD_PARAMETER(myPlist,'REPORT','foo.RDF');
        SRW.ADD_PARAMETER(myPlist,'USERID','foo/bar');
        SRW.ADD_PARAMETER(myPlist,'DISTRIBUTE','YES');
        SRW.ADD_PARAMETER(myPlist,'DESTINATION','filename.xml');
        SRW.ADD_PARAMETER(myPlist,'ExpenseID',:new.ExpID);
        myIdent := SRW.RUN_REPORT(myPlist);
    END IF;
END;
```

This is one way to move this kind of logic from your application into the database and use the database as a central storage for business processes.

> **Note:** You'll find additional examples of the Event-Driven Publishing API in action in the demo script `srw_test.sql`, included with your OracleAS Reports Services installation.

17.4 Integrating with Oracle Advanced Queuing

Oracle Advanced Queuing is a means for building an asynchronous request/response mechanism around a so-called queue and two processes: ENQUEUE, which puts MESSAGES into a queue, and DEQUEUE, which reads the queue.

Advanced queuing provides sophisticated mechanisms for distributing messages across queues and for queue subscription. These mechanisms are all built on top of these basic elements (ENQUEUE, DEQUEUE, and MESSAGES).

With the Event-Driven Publishing API you can use these queues to store and transmit report jobs. You can even build your own queuing mechanism if the one provided with OracleAS Reports Services does not fit your needs.

17.4.1 Creating a Queue That Holds Messages of Type SRW_PARAMLIST

A queue is a table in the database that holds, along with several administrative columns, an object column that represents a message. In our case the message is the parameter list.

The dbms_AQadm package, provided with Advanced Queuing, contains all the administrative functions required for setting up an advanced queuing system.

Use dbms_AQadm.Create_Queue_Table to create the physical table in the database. You must pass it a name for the table and a name for the object type that will define the message for this queue.

For example:

```
…
execute dbms_AQadm.Create_Queue_Table
    (queue_Table=>'queuename._tab',
    queue_Payload_Type=>'SRW_PARAMLIST_OBJECT',
    compatible=>'9.0');
```

In earlier examples, we created the object type SRW_PARAMLIST_OBJECT that encapsulates the SRW_PARAMLIST type in object notation so it can be used as a message.

After creating the queue table, you must create the queue with dbms_AQadm.Create_Queue and start the queue with dbms_AQadm.Start_Queue.

For example:

```
…
execute dbms_AQadm.Create_Queue
    (Queue_Name=>'queuename',Queue_Table=>'queuename._tab');
prompt … starting queue
execute dbms_AQadm.Start_Queue
    (Queue_Name=>'queuename');
…
```

> **Note:** You'll find a complete example for setting up, creating, and starting a simple queue in the demo file `srwAQsetup.sql`, included with your OracleAS Reports Services installation.

Having created and started the queue, what you need now is a procedure that creates a message in this queue and a procedure that reads out the queue and submits the job to the server. These are discussed in the following sections.

17.4.2 Creating the Enqueuing Procedure

The enqueuing procedure is responsible for putting a message into the queue. This procedure can be part of your application, called by a database-trigger, or provided via an external mechanism. In this section, we will provide an example of creating a stored procedure that puts a simple message in this queue.

Because our message is the parameter list itself, the procedure is fairly easy. We use the same code we used in earlier sections to create a parameter list. In addition to the variables we used, we define an `object` variable to hold the message we will put into the queue.

```
…
  plist_object SRW_ParamList_Object;
…
```

After creating the parameter list we create the actual message object using the object constructor.

```
…
plist_object := SRW_PARAMLIST_OBJECT(plist);
…
```

Then we enqueue the message using the enqueue procedure provided by Advanced Queuing.

```
…
dbms_aq.enqueue(queue_name => 'myQueue',
    enqueue_options => enqueue_options,
    message_properties => message_properties,
    payload => PList_Object,
    msgid => message_handle);
…
```

The message is put into the queue. Because we did not set up any message distribution, the message will stay in the queue until it is fetched by a dequeuing-procedure, which is discussed in the next section.

> **Note:** For the exact syntax of dbms_aq.enqueue refer to the Advanced Queuing API Reference document.
>
> You'll find additional examples in the srwAQsetup.sql file included with your OracleAS Reports Services installation.
>
> Look for upcoming information about Reports APIs on the Oracle Technology Network, (http://otn.oracle.com).

17.4.3 Creating the Dequeuing Procedure

A dequeuing procedure reads out all available messages in a queue and processes them. In our case, we want to read out the message and submit a job to the server using the parameter list that was attached to the message.

To accomplish this, we follow this example:

```
BEGIN
    dequeue_options.wait := 1;
loop
    DBMS_AQ.DEQUEUE(queue_name => 'myQueue',
        dequeue_options => dequeue_options,
        message_properties => message_properties,
        payload => PList_Object,
        msgid => message_handle);
    COMMIT;
    plist := plist_object.params;
    r_jid := SRW.RUN_REPORT(plist);
end loop;
exception when aq_timeout then
    begin
        NULL;
    end;
END;
```

This code example will read out the queue until all messages have been processed. Time allowed for processing is determined by the time-out defined in the second line of code. This time-out defines the amount of seconds the dequeue procedure should wait for a message before creating a time-out exception.

The `DBMS_AQ.DEQUEUE` built-in is provided by Advanced Queuing for reading out messages. It puts the payload of the message, the object that holds the information, into the object defined by the payload parameter.

Using `plist`, we extract the information from the payload object. As mentioned before, our object holds a parameter list. It is stored in the attribute `PARAMS` inside the object. The extracted parameter list is then handed over to `SRW.RUN_REPORT` for submitting the job.

If you want to avoid the need for invoking this dequeuing procedure by hand, you can run it as a job inside the database.

Part III

National Language Support and Bidirectional Support

Part III provides information about Reports-related National Language Support settings and bidirectional support. It includes the following chapter:

- Chapter 18, "Implementing NLS and Bidirectional Support"

18

Implementing NLS and Bidirectional Support

When you design reports to be deployed to different countries, you must consider such things as character sets and text reading order. OracleAS Reports Services includes the support you need to address any issues related to these considerations: National Language Support (NLS) for character sets and bidirectional support for text reading order.

Oracle NLS makes it possible to design applications that can be deployed in several different languages. Oracle supports most European, Middle Eastern, and Asian languages. NLS enables you to:

- Use international character sets (including multibyte character sets)

- Display data according to the appropriate language and territory conventions

- Extract strings that appear in your interface and translate them

Bidirectional support enables you to display data in either a left-to-right or right-to-left orientation, depending on the requirements of your audience.

This chapter provides a look at NLS architecture, including NLS settings relevant to Reports; explains how to specify character sets in a JSP; and offers information on bidirectional, Unicode, and translation support available through Oracle9*i*. It includes the following main sections:

- NLS Architecture

- NLS Environment Variables

- Specifying a Character Set in a JSP or XML File

- Bidirectional Support

- Unicode

- [Translating Applications](#)

18.1 NLS Architecture

Oracle NLS architecture consists of two parts:

- [Language-Independent Functions](#)
- [Language-Dependent Data](#)

18.1.1 Language-Independent Functions

Language-independent functions handle manipulation of data in an appropriate manner, depending on the language and territory of the runtime operator. Data is automatically formatted according to local date and time conventions.

18.1.2 Language-Dependent Data

With language-dependent data, you can isolate your data. This enables your application to deal only with translating strings that are unique to your application.

Because the language-dependent data is separate from the code, the operation of NLS functions is governed by the data supplied at runtime. New languages can be added and language-specific application characteristics can be altered without requiring code changes. This architecture also enables language-dependent features to be specified for each session.

18.2 NLS Environment Variables

NLS environment variables are automatically set to default values during Oracle Application Server installation.

> **Note:** On a given OracleAS Reports Services host machine, you can specify only one language. If you are providing application server services to a multilingual audience, you must have a separate host machine for each language.

Table 18–1 lists and describes NLS-related environment variables that are particularly relevant to OracleAS Reports Services.

> **Note:** For more information on all NLS environment variables, see the *Oracle Application Server 10g Globalization Guide* on the Oracle Technology Network, (`http://otn.oracle.com`).

Table 18–1 Environment variables particularly related to OracleAS Reports Services

Variable	Description
`NLS_LANG`	Relevant to OracleAS Reports Services. The language settings used by OracleAS Reports Services.
`DEVELOPER_NLS_LANG`	The language for Reports Builder.
`USER_NLS_LANG`	The language for the Reports Runtime component.

18.2.1 NLS_LANG Environment Variable

The `NLS_LANG` environment variable specifies the language, territory, and character set settings to be used by OracleAS Reports Services. Specifically:

- The language for messages displayed to the user

- The default format masks used for `DATE` and `NUMBER` data types

- The sorting sequence

- The character set

> **Note:** This environment variable is set automatically when you install Oracle Application Server. Refer to Defining the NLS_LANG Environment Variable for more information about changing the environment variable after installing Oracle Application Server.

The syntax for `NLS_LANG` is:

```
NLS_LANG=language_territory.charset
```

The values are defined as follows:

- `language`

  Specifies the language and its conventions for displaying messages (including error messages) as well as day and month names. If language is not specified, then the value defaults to American.

- ■ `territory`

  Specifies the territory and its conventions for default date format, decimal character used for numbers, currency symbol, and calculation of week and day numbers. If territory is not specified, then the value defaults to America.

- ■ `charset`

  Specifies the character set in which data is displayed. This should be a character set that matches your language and platform. This option also specifies the character set used for displaying messages.

> **Note:** When you use features like OracleAS Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by OracleAS Reports Services may override the initial `NLS_LANG` setting. This change may in turn affect the behavior of the running report, such as bidirectional output in PDF. On UNIX platforms, you can work around this issue by setting the `NLS_LANG` explicitly in `report.sh`. You can also use the new environment switching functionality to dynamically set the environment for reports. Refer to Dynamic Environment Switching, for more information.

Table 18–2 lists commonly used language, territory, and character values for `NLS_LANG`:

Table 18–2   Commonly used `NLS_LANG` *values*

Language	Language_Territory.Character Set
American	AMERICAN_AMERICA.US7ASCII
Arabic	ARABIC_UNITED ARAB EMIRATES.AR8ISO8859P6
Brazilian Portuguese	BRAZILIAN PORTUGUESE_BRAZIL.WE8DEC
Bulgarian	BULGARIAN_BULGARIA.CL8ISO8859P5
Canadian French	CANADIAN FRENCH_CANADA.WE8ISO8859P1
Catalan	CATALAN_CATALONIA.WE8ISO8859P1
Croatian	CROATIAN_CROATIA.EE8ISO8859P2
Czech	CZECH_CZECH REPUBLIC.WE8ISO8859P1
Danish	DANISH_DENMARK.WE8ISO8859P1

Table 18–2   Commonly used `NLS_LANG` *values*

Language	Language_Territory.Character Set
Dutch	DUTCH_THE NETHERLANDS.WE8ISO8859P1
Egyptian	ARABIC_UNITED ARAB EMIRATES.AR8ISO8859P6
English (American)	See American
English (United Kingdom)	ENGLISH_UNITED KINGDOM.WE8DEC
Estonian	ESTONIAN_ESTONIA.BLT8MSWIN1257
Finnish	FINNISH_FINLAND.WE8ISO8859P1
French	FRENCH_FRANCE.WE8ISO8859P1
German	GERMAN_GERMANY.WE8ISO8859P1
Greek	GREEK_GREECE.EL8ISO8859P7
Hebrew	HEBREW_ISRAEL.IW8ISO8859P8
Hungarian	HUNGARIAN_HUNGARY.EE8ISO8859P2
Icelandic	ICELANDIC_ICELAND.WE8ISO8859P1
Indonesian	INDONESIAN_INDONESIA.WE8ISO8859P1
Italian	ITALIAN_ITALY.WE8DEC
Japanese	JAPANESE_JAPAN.JA16EUC
Korean	KOREAN_KOREA.KO16KSC5601
Latin America Spanish	LATIN AMERICAN SPANISH_AMERICA.WE8DEC
Latvian	LATVIAN_LATVIA.NEE8ISO8859P4
Lithuanian	LITHUANIAN_LITHUANIA.NEE8ISO8859P4
Mexican Spanish (see also Spanish)	MEXICAN SPANISH_MEXICO.WE8DEC
Norwegian	NORWEGIAN_NORWAY.WE8DEC
Polish	POLISH_POLAND.EE8ISO8859P2
Portuguese	PORTUGUESE_PORTUGAL.WE8DEC
Romanian	ROMANIAN_ROMANIA.EE8ISO8859P2
Russian	RUSSIAN_CIS.RU8PC855
Simplified Chinese	SIMPLIFIED CHINISE_CHINA.ZHS16CGB231280

Table 18–2   Commonly used `NLS_LANG` *values*

Language	Language_Territory.Character Set
Slovak	SLOVAK_SLOVAKIA.EE8ISO8859P2
Spanish (see also Mexican Spanish)	SPANISH_SPAIN.WE8DEC
Swedish	SWEDISH_SWEDEN.WE8DEC
Thai	THAI_THAILAND.TH8TISASCII
Traditional Chinese	TRADITIONAL CHINESE_TAIWAN.ZHT32EUC
Turkish	TURKISH_TURKEY.WE8ISO8859P9
Ukrainian	UKRAINIAN_UKRAINE.CL8ISO8859P5
Vietnamese	VIETNAMESE_VIETNAM.VN8VN3

Your `NLS_LANG` setting should take into account regional differences between countries that use (basically) the same language. For example, if you want to run in French (as used in France), then you set the `NLS_LANG` environment variable:

```
NLS_LANG=FRENCH_FRANCE.WE8ISO8859P1
```

If you want to run in French, but this time as used in Switzerland, you would set the `NLS_LANG` environment variable:

```
NLS_LANG=FRENCH_SWITZERLAND.WE8ISO8859P1
```

> **Note:**   The language for the `rwserver` pages such as `help`, `showjobs`, `showenv` and error messages are delivered from machine's locale (or `LANG` on Solaris) and not `NLS_LANG`. For example, you have set your middle tier locale to French and `NLS_LANG=JAPANESE_JAPAN.JA16SJIS`, the `showjobs` or error messages will be displayed in French, not in Japanese.

18.2.1.1 Defining the NLS_LANG Environment Variable

You define the `NLS_LANG` environment variable in the same way you define other environment variables on your Windows or UNIX operating system.

18.2.1.1.1 **Windows** To define the `NLS_LANG` environment variable on Windows, do the following:

1. Open the Windows registry.

> **Note:** Back up your registry before you edit it.

2. Expand the **HKEY_LOCAL_MACHINE** node, then expand the **SOFTWARE** node.

3. Click **ORACLE** to display the Oracle environment variables in the right panel of the Registry Editor.

4. Double-click the `NLS_LANG` environment variable.

5. Type the new value for `NLS_LANG` in the **Value** data text box.

6. Click **OK**.

18.2.1.1.2 **UNIX** To define the `NLS_LANG` environment variable on the UNIX platform, set it in the shell script `reports.sh`, located in your *ORACLE_HOME*/bin directory.

18.2.1.2 Character Sets

The character set component of the NLS environment variables specifies the character set in which data is represented in your environment. When data is transferred from a system using one character set to a system using another character set, it is processed and displayed correctly on the second system, even though some characters might be represented by different binary values in the character sets.

18.2.1.2.1 **Character Set Design Considerations** If you are designing a multilingual application, or even a single-language application that runs with multiple character sets, you need to determine the character set most widely used at runtime and then generate with the NLS environment variable set to that particular character set.

If you design and generate an application in one character set and run it in another character set, performance can suffer. Furthermore, if the runtime character set does not contain all the characters in the generate character set, then question marks appear in place of the unrecognized characters.

Portable Document Format (PDF) supports multibyte character sets.

18.2.1.2.2   Font Aliasing on Windows Platforms There might be situations where you create an application with a specific font but find that a different font is being used when you run that application. You would most likely encounter this when using an English font (such as MS Sans Serif or Arial) in environments other than Western European. This occurs because OracleAS Reports Services checks to see if the character set associated with the font matches the character set specified by the language environment variable. If the two do not match, OracleAS Reports Services automatically substitutes the font with another font whose associated character set matches the character set specified by the language environment variable. This automatic substitution assures that the data being returned from the database gets displayed correctly in the application.

> **Note:**   If you enter local characters using an English font, then Windows does an implicit association with another font.

There might be cases, however, where you do not want this substitution to take place. You can avoid this substitution by mapping all desired fonts to the WE8ISO8859P1 character set in the font alias file (uifont.ali). For example, if you are unable to use the Arial font in your application, you can add the following line to your font alias file (located at *ORACLE_HOME*\TOOLS\COMMON\):

```
ARIAL.....=ARIAL.....WE8ISO8859P1
```

Each line in the uifont.ali file takes the following syntax:

```
Face.Size.Style.Weight.Width.CharSet=Face.Size.Style.Weight.Width.CharSet
```

In this example, you're saying that any ARIAL font should be mapped to the same value, but with the WE8ISO8859P1 character set.

Refer to NLS Environment Variables for more information about the language environment variables.

18.2.1.3 Language and Territory

While the character set ensures that the individual characters needed for each language are available, support for national conventions provides correct localized display of data items.

The specified language determines the default conventions for the following characteristics:

- Language for server messages

- Language for day and month names and their abbreviations (specified in the SQL functions `TO_CHAR` and `TO_DATE`)

- Symbol equivalents for AM, PM, AD, and BC

- Default sorting sequence for character data when `ORDER BY` is specified (`GROUP BY` uses a binary sort unless `ORDER BY` is specified)

- Writing direction (both right to left and left to right)

- Affirmative and negative response strings

For example, if the language is set to French, then the following messages in English are converted to French:

```
English:
ORA-00942: table or view does not exist
FRM-10043: Cannot open file.

French:
ORA-0092: table ou vue inexistante
FRM-10043: Ouverture de fichier impossible
```

The specified territory determines the conventions for the following default date and numeric formatting characteristics:

- Date format

- Decimal character and group separator

- Local currency symbol

- ISO currency symbol

- Week start day

- Credit and debit symbol

- ISO week flag

- List separator

For example, if the territory is set to France, then the numbers are formatted using a comma as the decimal character.

18.2.2 DEVELOPER_NLS_LANG and USER_NLS_LANG Environment Variables

If you must use two sets of resource and message files at the same time, then two other language environment variables are available. These can be used after Oracle Application Server installation is completed.

■   DEVELOPER_NLS_LANG

■   USER_NLS_LANG

The syntax for DEVELOPER_NLS_LANG and USER_NLS_LANG is the same as for the NLS_LANG environment variable. That is:

```
DEVELOPER_NLS_LANG=language_territory.charset
USER_NLS_LANG=language_territory.charset
```

Use these environment variables in lieu of the NLS_LANG environment variable in the following situations:

■   You prefer to use the Reports Builder in English, but you are developing an application for another language. DEVELOPER_NLS_LANG and USER_NLS_LANG environment variables allow you to use different language settings for the Reports Builder and Reports Runtime.

■   You are creating an application to run in a language for which a local language version of Reports Builder is not currently available.

If these environment variables are not specifically set, then NLS_LANG default values will be used.

18.3 Specifying a Character Set in a JSP or XML File

If you are producing HTML with your JSP, then you may need to add a character set to your JSP file using the following syntax (this one specifies a Japanese character set):

```
<META http-equiv="Content-Type" content="text/html;charset=shift_jis">
```

Additionally, if you plan on outputting a report to XML, you may wish to include a character set in the report's XML Prolog Value property, following this syntax:

```
<?xml version="1.0" encoding="shift_jis" ?>
```

> **Note:** You can define a character set encoding when saving a report to an XML file. For more information, refer to REPORTS_NLS_XML_CHARSETS.

In both instances, the values expressed for the character set should call a character set that is compatible with the one specified for the host environment. The values for character sets used on the Web are different from the values expressed in the NLS_LANG environment variable. Table 18–3 lists commonly used values for the charset or encoding parameter:

> **Note:** The values for charset and encoding are not case sensitive. You can enter them in lower- or uppercase.

> **Note:** to set the character set in a .rdf file that you plan to use to generate HTML, you must ensure that the Before Report Escape property includes the following:
>
> `charset="text/html;charset=&encoding"`
>
> &encoding is then replaced at runtime with the appropriate setting.

Table 18–3   Valid values for a charset or encoding parameter

Language	Valid Character Set(s)
Afrikaans	iso-8859-1, windows-1252
Albanian	iso-8859-1, windows-1252
Arabic	iso-8859-6
Basque	iso-8859-1, windows-1252
Bulgarian	iso-8859-5
Byelorussian	iso-8859-5
Catalan	iso-8859-1, windows-1252
Croatian	iso-8859-2

Table 18–3   Valid values for a charset or encoding parameter

Language	Valid Character Set(s)
Czech	iso-8859-2
Danish	iso-8859-1, windows-1252
Dutch	iso08859-1, windows-1252
English	iso-8859-1, windows-1252
Esperanto	iso-8859-3 (not widely supported in browsers)
Estonian	iso-8859-15
Faroese	iso-8859-1, windows-1252
Finnish	iso-8859-1, windows-1252
French	iso-8859-1, windows-1252
Galician	iso-8859-1, windows-1252
German	iso-8859-1, windows-1252
Greek	iso-8859-1
Hebrew	iso-8859-8
Hungarian	iso-8859-2
Icelandic	iso-8859-1, windows-1252
Inuit languages	iso-8859-10 (not widely supported in browsers)
Irish	iso-8859-1
Italian	iso-8859-1
Japanese	shift_jis, iso-2202-jp, euc-jp
Korean	euc-kr
Lapp	iso-8859-10 (not widely supported in browsers)
Latvian	iso-8859-13, windows-1257
Lithuanian	iso-8859-13, windows-1257
Macedonian	iso-8859-5
Maltese	iso-8859-3 (not widely supported in browsers)
Norwegian	iso-8859-1, windows-1252
Polish	iso-8859-2

Table 18–3   Valid values for a charset or encoding parameter

Language	Valid Character Set(s)
Portuguese	iso-8859-1, windows-1252
Romanian	iso-8859-2
Russian	koi-8-r, iso-8859-5
Scottish	iso-8859-1, windows-1252
Serbian	iso-8859-5
Slovak	iso-8859-2
Slovenian	iso-8859-2
Spanish	iso-8859-1, windows-1252
Swedish	iso-8859-1, windows-1252
Turkish	iso-8859-9, windows-1254
Ukrainian	iso-8859-5

18.4 Bidirectional Support

Bidirectional support enables you to design applications in Middle Eastern and North African languages whose natural writing direction is right to left. Bidirectional support enables you to control:

- Layout direction, which includes displaying items with labels at the right of the item and correct placement of check boxes and radio buttons

- Reading order, which includes text direction (e.g., right to left or left to right)

- Alignment, which includes switching point-of-origin from upper left to upper right

- Initial keyboard state, which controls whether local or Roman characters are produced automatically when the user begins data entry in forms (the end user can override this setting)

When you are designing bidirectional applications, you might want to use the NLS environment variables DEVELOPER_NLS_LANG and USER_NLS_LANG rather than NLS_LANG. For example, if you want to use an American interface while developing an Arabic application in a Windows environment, then set these environment variables as follows:

```
DEVELOPER_NLS_LANG=AMERICAN_AMERICA.AR8MSWIN1256
```

```
USER_NLS_LANG=ARABIC_UNITED ARAB EMIRATES.AR8MSWIN1256
```

Note that, in this example, the DEVELOPER_NLS_LANG environment variable uses an Arabic character set. Refer to NLS Environment Variables for more information about environment variables.

> **Note:** When you use features like OracleAS Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by OracleAS Reports Services may override the initial NLS_LANG setting. This change may in turn affect the behavior of the running report, such as bidirectional output in PDF. On UNIX platforms, you can work around this issue by setting the NLS_LANG explicitly in report.sh. You can also use the new environment switching functionality to dynamically set the environment for reports. Refer to Dynamic Environment Switching, for more information.

18.5 Unicode

Unicode is a global character set that allows multilingual text to be displayed in a single application. This enables multinational corporations to develop a single multilingual application and deploy it worldwide.

Global markets require a character set that:

- Allows a single implementation of a product for all languages, yet is simple enough to be implemented everywhere

- Contains all major living scripts

- Supports multilingual users and organizations

- Enables worldwide interchange of data via the Internet

18.5.1 Unicode Support

OracleAS Reports Services provides Unicode support. If you use Unicode, you are able to display multiple languages, both single-byte languages such as Western Europe, Eastern Europe, Bidirectional Middle Eastern, and multibyte Asian languages such as Chinese, Japanese, and Korean (CJK) in the same application.

Use of a single character set that encompasses all languages eliminates the need to have various character sets for various languages. For example, to display a

multibyte language such as Japanese, the NLS_LANG environment variable must be set to the following:

```
NLS_LANG=JAPAN_JAPANESE.JA16SJIS
```

To display a single-byte language such as German, NLS_LANG must be set to the following:

```
NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1
```

The obvious disadvantage of this scheme is that applications can only display characters from one character set at a time. Mixed character set data is not possible.

With the Unicode character set, you can set the character set portion of NLS_LANG to UTF8 instead of a specific language character set. This allows characters from different languages and character sets to be displayed simultaneously. For example, to display Japanese and German together on the screen, the NLS_LANG variable must be set to one of the following:

```
NLS_LANG=JAPAN_JAPANESE.UTF8
```

```
NLS_LANG=GERMAN_GERMANY.UTF8
```

Unicode capability gives the application developer and end user the ability to display multilingual text in a form. This includes text from a database containing Unicode, multilingual text, text in graphical user interface (GUI) objects (for example, button labels), text input from the keyboard, and text from the clipboard. OracleAS Reports Services currently supports Unicode on Windows.

> **Note:** If you develop applications for the Web, then you can use Unicode because of the Unicode support provided by Java through the browser.

18.5.2 Unicode Font Support

OracleAS Reports Services relies on the operating system for the font and input method for different languages. To enter and display text in a particular language, you must be running a version of the operating system that supports that language. Font support is limited but not restricted to the operating system font.

Windows NT release 4.0 and later versions provide True Type Big Fonts. These fonts contain all the characters necessary to display or print multilingual text. If you try to type, display, or print multilingual text and see unexpected characters, then you

are probably not using a Big Font. Big Fonts provided by Microsoft under Windows NT release 4.0 and later, are as follows:

- Arial

- Courier New

- Lucida Console

- Lucida Sans Unicode

- Times New Roman

Third-party Unicode fonts are also available.

18.5.3 Enabling Unicode Support

To enable Unicode support, set the NLS_LANG environment variable as follows:

```
NLS_LANG=language_territory.UTF8
```

Refer to NLS Environment Variables for more information about environment variables.

18.5.4 Using ALTER SESSION

You can use the SQL command ALTER SESSION to override some NLS defaults, if you are connected to an Oracle database via the USERID keyword or have connected via the Connect dialog in Reports Builder. For example, suppose you are building a report against an Oracle database that will publish data to different geographic locations. You might want to change the currency symbol, thousands grouping, and decimal indicator that the Oracle database uses when it formats a currency field depending on a user parameter. You could accomplish this task in several ways, but one method is to alter the Oracle database session NLS_LANG variable in a Before Report trigger.

> **Note:** ALTER SESSION does not apply to pluggable data source queries (for example, JDBC and XML).

> **Note:** For more information on the ALTER SESSION command, see the *Oracle9i SQL Reference*, available on the Oracle Technology Network, (`http://otn.oracle.com`).

18.6 Translating Applications

In any OracleAS Reports Services application, you see many types of messages, including:

- Error messages from the database

- Runtime error messages produced by OracleAS Reports Services

- Messages and boilerplate text defined as part of the application

If the NLS environment variable is set correctly and the appropriate message files are available, then translation of messages for the first two items is done for you. To translate messages and boilerplate text defined as part of the application, you can use the Oracle translation tool, TranslationHub, and you might also find it useful to use PL/SQL Libraries for strings of code.

> **Note:** You'll find information about using TranslationHub on the Oracle9iDS documentation CD and on the Oracle Technology Network, (`http://otn.oracle.com`).

Manual translation is required for constant text within a PL/SQL block because that text is not clearly delimited, but is often built up from variables and pieces of strings. To translate these strings, you can use PL/SQL libraries to implement a flexible message structure.

You can use attachable PL/SQL libraries to implement a flexible message function for messages that are displayed programmatically by the built-in routine SRW.MESSAGE, or by assigning a message to a display item from a trigger or procedure. The library can be stored on the host and dynamically attached at runtime. At runtime, based on a search path, you can pull in the attached library. For example, a library might hold only the Italian messages:

```
FUNCTION nls_appl_mesg(inexe_no NUMBER)
RETURN CHAR
IS
   msg CHAR(80):
BEGIN
   IF      index_no = 1001 THEN
      msg := 'L' 'impiegato che Voi cercate non esiste...';
   ELSEIF  index_no = 1002 THEN
```

```
      msg := 'Lo stipendio non puo essere minore di zero.':
   ELSEIF  ...
   .
   .
   .
   ELSE
      msg := 'ERRORE: Indice messaggio inesistente.';
   END IF;
   RETURN msg;
END;
```

A routine like this could be used anywhere a character expression would normally be valid. For example, to display text with the appropriately translated application message, you might include the following code:

```
Change_Alert_Message('My_Error_Alert', nls_appl_mesg(1001));
n := Show_Alert('My_Error_Alert');
```

To change the application to another language, simply replace the PL/SQL library containing the nls_appl_mesg function with a library of the same name containing the nls_appl_mesg function with translated text.

Part IV

Performance

Part IV provides information on managing, monitoring, and tuning your Oracle Application Server Reports Services environment. It includes the following chapters:

19

Managing and Monitoring OracleAS Reports Services

Oracle Enterprise Manager, included with Oracle Application Server, provides managing and monitoring services for OracleAS Reports Services.

- The Oracle Enterprise Manager Application Server Control provides both management and monitoring functions for your Reports Servers.

This chapter describes the managing and monitoring capabilities of Oracle Enterprise Manager as they relate to OracleAS Reports Services and tells you how to configure the Reports Server with Oracle Enterprise Manager. It includes the following main sections:

- Configuring the Reports Server for Oracle Enterprise Manager
- Navigating to the Reports Server Page
- Starting, Stopping, and Restarting Reports Servers
- Viewing and Managing Job Queues
- Monitoring Server Performance
- Viewing and Changing the Reports Server Configuration Files
- Viewing and Linking to Server Cluster Members
- Viewing Port Numbers
- Changing Your Middle-Tier

> **Note:** The Reports Server pages in Oracle Enterprise Manager
> include context sensitive online help topics that offer information
> about the items that appear on each page. In Oracle Enterprise
> Manager, click the **Help** link to display help.

> **See Also:** For more information on Oracle Enterprise Manager,
> *Oracle Application Server 10g Administrator's Guide*, available on the
> Oracle Application Server documentation CD.

19.1 Configuring the Reports Server for Oracle Enterprise Manager

Reports Servers are automatically configured in OPMN and registered with Oracle
Enterprise Manager during installation of Oracle Application Server. If you add any
Reports Servers after installing Oracle Application Server, you must register the
new server(s) in:

- The Oracle Enterprise Manager's `targets.xml` file. Refer to Configuring
  Reports Server with the Oracle Process Manager and Notification Server and
  Oracle Enterprise Manager for more information on the `targets.xml` file and
  its usage.

- The Oracle Process Manager and Notification Server's `opmn.xml` file. Refer to
  `opmn.xml` for more information on the `opmn.xml` file and its usage.

> **Note:** Use `addNewServerTarget.sh` (UNIX) or
> `addNewServerTarget.bat` (Windows) to add a Reports Server
> target to `targets.xml` and `opmn.xml`.

> **See Also:** For more information:
>
> - Configuring Reports Server with the Oracle Process Manager
>   and Notification Server and Oracle Enterprise Manager

19.2 Navigating to the Reports Server Page

This section describes how to get to the Reports Server page:

- Navigating to the Reports Server Page in the Application Server Control

19.2.1 Navigating to the Reports Server Page in the Application Server Control

You can get to the Reports Server page in a variety of ways. This procedure is just one possible way.

1. Launch the Application Server Control. For more information on how to launch the console, refer to your Oracle Enterprise Manager documentation.

2. The console opens on the Farm page as shown in Figure 19–1, which displays the available Oracle Application Server instances.

Figure 19–1   Farm Page in Oracle Enterprise Manager

3. Click the Oracle Application Server instance that contains the Reports Server you want to view. This link takes you to the page for that instance as shown in Figure 19–2.

Figure 19–2   Oracle Application Server Page in Oracle Enterprise Manager
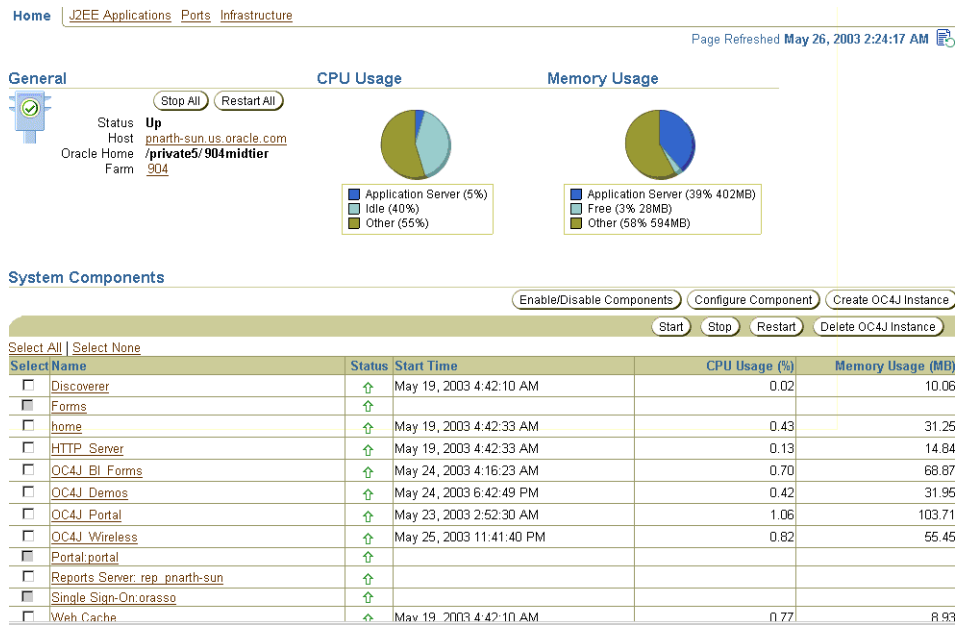
4. Choose the Reports Server you wish to view from the list of components. This link takes you to the main page for the Reports Server, which should look similar to the one in Figure 19–3.

Figure 19–3   Reports Server Main Page in Oracle Enterprise Manager

19.3 Starting, Stopping, and Restarting Reports Servers

To start, stop, or restart a Reports Server:

1. Navigate to the Reports Server main page in the Application Server Control for the Reports Server you want to start, stop, or restart.

   > **Note:**   You can only start, stop, and restart the Reports Server from the Application Server Control.

   > **See Also:**   Navigating to the Reports Server Page in the Application Server Control for more information.

2. On the Reports Server's main page, you should see the following:

   - Click the Start button to start the server.

   - Click the Stop button to stop the server.

■ Click the Restart button to restart the server.

These buttons appear on a Reports Server main page according to the server's current state:

■ When the server is down, the Stop (disabled) and Start buttons display.

■ When the server is up, the Stop and Restart buttons display.

> **Note:** Refer to the Oracle Enterprise Manager Application Server Control section for more information on the various tabs displayed on the Reports Server home page.

19.4 Viewing and Managing Job Queues

You can view and manage your job queues in the Application Server Control.

■ Viewing and Managing Job Queues in the Application Server Control

> **Note:** If you want to manage the job queue (e.g., cancel jobs or rerun jobs), you must do so from the Application Server Control. To get to the Application Server Control, return to the Reports Server main page and click **Administration** under **Related Links**. Once there, follow the instructions in Viewing and Managing Job Queues in the Application Server Control.

19.4.1 Viewing and Managing Job Queues in the Application Server Control

The following sections describe:

■ Viewing and Managing the Current Jobs Queue

■ Viewing and Managing the Scheduled Jobs Queue

■ Viewing and Managing the Finished Jobs Queue

■ Viewing and Managing the Failed Jobs Queue

19.4.1.1 Viewing and Managing the Current Jobs Queue

The reports Current Jobs queue lists all jobs currently running on a particular Reports Server.

Use Oracle Enterprise Manager for:

- Viewing a Reports Server's Current Jobs Queue

- Cancelling a Current Job

19.4.1.1.1 Viewing a Reports Server's Current Jobs Queue To view a current jobs queue:

1. Navigate to the Reports Server main page in Oracle Enterprise Manager for the Reports Server you want to view.

> **See Also:** Navigating to the Reports Server Page for more information.

2. On the Reports Server's main page, scroll down to the **Jobs** section and click the number next to **Current Jobs**.

If there are no current jobs, a zero appears next to **Current Jobs** and there will be no link because the queue is empty.

Table 19–1 lists and describes information provided in the current jobs queue.

Table 19–1   Information provided in the Current Jobs Queue

Item	Description
Select	Use this radio button to select a particular job. On the Current Jobs Queue page, this function is most useful when you wish to cancel a job. Click the **Select** radio button next to a job you wish to cancel, then click the **Cancel** button near the top of the page.
Id	This displays a unique job identifier assigned to this job by the Reports Server. This number is strictly under the server's control and cannot be reset by a user.
Job Name	If you specified a job name in the command line you used to run this job, that name is listed here; otherwise, it is the name of the job provided for the "report=" or "module=" parameter of the job request.
Owner	This displays the user ID under which this job is running.
Output Type	Lists the destination type (destype) specified for this job at runtime.
Output Format	Lists the output format (desformat) specified for this job at runtime.
Queued At	Lists the date and time this request was placed in the job queue.

Table 19–1   Information provided in the Current Jobs Queue

Item	Description
Started At	Lists the date and time when this job was run.
Status	Status of the current job.

19.4.1.1.2   Cancelling a Current Job   To cancel a current job:

1. On the **Current Jobs** page, click the **Select** radio button next to the job you want to cancel.

2. Click the **Cancel** button. This button does not display in Oracle Enterprise Manager if there are no currently running jobs.

If you wish to rerun the job, you can do so from the Finished Jobs page.

19.4.1.2 Viewing and Managing the Scheduled Jobs Queue

The scheduled jobs queue lists all jobs scheduled to run on a particular Reports Server.

Use Oracle Enterprise Manager for:

- Viewing a Reports Server's Scheduled Jobs Queue
- Cancelling a Scheduled Job

19.4.1.2.1   Viewing a Reports Server's Scheduled Jobs Queue   To view a scheduled jobs queue:

1. Navigate to the Reports Server main page in Oracle Enterprise Manager for the Reports Server you want to view.

   See Also: Navigating to the Reports Server Page for more information.

2. On the Reports Server main page, scroll down to the **Jobs** section and click the number next to **Scheduled Jobs**.

   If there are no scheduled jobs, a zero appears next to **Scheduled Jobs** and there will be no link because the queue is empty.

Table 19–2 lists and describes information provided in the scheduled jobs queue.

Table 19–2 Information provided in the Scheduled Jobs Queue

Item	Description
Select	Use this radio button to select a particular job. On the Scheduled Job Queue page, this function is most useful when you wish to cancel a job. Click the **Select** radio button next to a job you wish to cancel, then click the **Cancel** button near the top of the page.
Id	This displays a unique job identifier assigned to this job by the Reports Server. This number is strictly under the server's control and cannot be reset by a user.
Job Name	If you specified a job name in the command line you used to run this job, that name is listed here; otherwise, it is the name of the job provided for the "report=" or "module=" parameter of the job request.
Owner	This displays the user ID under which this job is scheduled to run.
Output Type	Lists the destination type (destype) specified for this job.
Output Format	Lists the output format (desformat) specified for this job.
Repeat Interval	The frequency at which the job will be run, for example, daily, monthly, and so on.
Last Run At	The time stamp when the scheduled job was previously run.
Next Run At	The time stamp when the scheduled job was run immediately next.

19.4.1.2.2 Cancelling a Scheduled Job To cancel a scheduled job:

1. On the Scheduled Jobs page, click the **Select** radio button next to the job you want to cancel.

2. Click the **Cancel** button.

   This button does not display in Oracle Enterprise Manager if there are no scheduled jobs.

If you wish to rerun the job, you can do so from the Finished Jobs page.

19.4.1.3 Viewing and Managing the Finished Jobs Queue
The finished jobs queue lists all successfully completed jobs on a particular Reports Server.

Use Oracle Enterprise Manager for:

- Viewing a Reports Server's Finished Jobs Queue
- Viewing a Job's Trace File
- Viewing a Result from Cache
- Rerunning a Finished Job

19.4.1.3.1 Viewing a Reports Server's Finished Jobs Queue To view a finished jobs queue:

1. Navigate to the Reports Server main page in Oracle Enterprise Manager for the Reports Server you want to view.

   > **See Also:** Navigating to the Reports Server Page for more information.

2. On the Reports Server main page, scroll down to the **Jobs** section and click the number next to **Finished Jobs**.

   If there are no finished jobs, a zero appears next to **Finished Jobs** and there will be no link because the queue is empty.

Table 19–3 lists and describes information provided in the finished jobs queue.

Table 19–3   Information provided in the Finished Jobs Queue

Item	Description
Select	Use this radio button to select a particular job. On the Finished Job Queue page, this function is most useful for selecting a job and:
	Viewing its output
	Click the **Select** radio button next to a job you want to view, then click the **View Result** button near the top of the page.
	Viewing its trace results, provided that you included a trace command in the runtime command.
	Click the **Select** radio button next to a job with trace results you want to view, then click the **View Trace** button near the top of the page. Reports Server displays the No Log File Found error if the trace file is not found. Refer to trace for more information on how to specify the trace options for a job.
	Rerunning it
	Click the **Select** radio button next to a job you want to rerun, then click the **Rerun Report** button near the top of the page.
Id	Use this link to view the trace files for the job. This displays a unique job identifier assigned to this job by the Reports Server. This number is strictly under the server's control and cannot be reset by a user.
Job Name	Use this link to view the results of the job. If you specified a job name in the command line you used to run this job, that name is listed here; otherwise, it is the name of the job provided for the "report=" or "module=" parameter of the job request.
Owner	This displays the user ID under which this job was run.
Output Type	Lists the destination type (destype) specified for this job.
Output Format	Lists the output format (desformat) specified for this job.
Queued At	Lists the date and time this request was placed in the job queue.
Started At	Lists the date and time this job started running.
Finished At	Displays the date and time this job completed.
Status	Displays the finished status of the job. In the Finished Jobs Queue, status is always Finished Successfully.

> **Note:** On the Finished Jobs page, you can view the finished job report by using the **View Result** button for the selected Job **Id**. However, if you view the output of a job that has images in it using a Netscape browser, the job output is shown but the images are broken. You must use Internet Explorer to view the images correctly. This workaround is applicable only to Finished Jobs with images in it.

19.4.1.3.2 Viewing a Job's Trace File The trace file contains tracing information about the report's execution. You set which tracing options you want the Reports Server to use in the job command line via the `traceopts` argument.

> **See Also:** View trace for more information about available tracing options.

To view a job's trace file:

1. In the **Select** column on a Finished Jobs page, click the radio button next to the finished job whose trace file you want to view.

2. Click the **View Trace** button near the top of the page. Alternatively, you can click the ID link.

> **Note:** Reports Server displays the `No Log File Found` error if the trace file is not found. Refer to trace for more information on how to specify the trace options for a job.

19.4.1.3.3 Viewing a Result from Cache When monitoring you Reports Server, it can sometimes be useful to quickly check a report's output to verify that it produced the desired output. To view a job result from the Report Server cache:

1. In the **Select** column on a Finished Jobs page, click the radio button next to the finished job you want to view.

2. Click the **View Result** button near the top of the page. Alternatively, you can click the Name link.

   The result opens in a separate browser window.

19.4.1.3.4   Rerunning a Finished Job   In some cases, you may want to quickly resubmit a report for execution (e.g., to refresh the contents of the cache). To rerun a job from the Finished Jobs page:

1. In the **Select** column, click the radio button next to the finished job that you want to rerun.

2. Click the **Rerun Report** button near the top of the page.

19.4.1.4 Viewing and Managing the Failed Jobs Queue

The failed jobs queue lists all jobs that were cancelled or terminated with errors on a particular Reports Server.

Use Oracle Enterprise Manager for:

- Viewing a Reports Server's Failed Jobs Queue

- Viewing a Failed Job's Trace File

- Rerunning a Failed Job

19.4.1.4.1   Viewing a Reports Server's Failed Jobs Queue   To view a failed jobs queue:

1. Navigate to the Reports Server main page in Oracle Enterprise Manager for the Reports Server you want to view.

   > **See Also:**   Navigating to the Reports Server Page for more information.

2. On the Reports Server's main page, scroll down to the **Jobs** section and click the number next to **Failed Jobs**.

   If there are no failed jobs, a zero appears next to **Failed Jobs** and there will be no link because the queue is empty.

Table 19–4 lists and describes information provided in the failed jobs queue.

Table 19–4   Information provided in the Failed Jobs Queue

Item	Description
Select	Use this radio button to select a particular job. On the Failed Jobs Queue page, this function is most useful for selecting a job and:
	Viewing its trace results, provided that you included a trace command in the runtime command.
	Click the **Select** radio button next to a job with trace results you want to view, then click the **View Trace** button near the top of the page.
	Rerunning it
	Click the **Select** radio button next to a job you want to rerun, then click the **Rerun Report** button near the top of the page.
Id	This displays a unique job identifier assigned to this job by the Reports Server. This number is strictly under the server's control and cannot be reset by a user.
Job Name	If you specified a job name in the command line you used to run this job, that name is listed here; otherwise, it is the name of the job provided for the "report=" or "module=" parameter of the job request.
Owner	This displays the user ID under which this job was run.
Output Type	Lists the destination type (destype) specified for this job.
Output Format	Lists the output format (desformat) specified for this job.
Queued At	Lists the date and time this request was placed in the job queue.
Started At	Lists the date and time this job started running.
Finished At	Displays the date and time this job was cancelled or terminated with error.
Status	Displays the finished status of the job. It informs you whether a job was cancelled by the user or terminated with error. In instances where a job was terminated with error, a brief error message is provided to indicate the cause of termination.

19.4.1.4.2 Viewing a Failed Job's Trace File To aid you in determining why a job failed, you can view the trace file for the failed job. To view a failed job's trace file:

1. In the **Select** column on a Failed Jobs page, click the radio button next to the failed job whose trace file you want to view.

2. Click the **View Trace** button near the top of the page. Alternatively, you can click the ID link.

> **Note:** Reports Server displays the `No Log File Found` error if the trace file is not found. Refer to trace for more information on how to specify the `trace` options for a job.

19.4.1.4.3 Rerunning a Failed Job In order to quickly check whether the problem with a failed job has been resolved, you can rerun the job from the Failed Jobs page. To rerun a failed job:

1. In the **Select** column on the Failed Jobs page, click the radio button next to the failed job you want to rerun.

2. Click the **Rerun Report** button near the top of the page.

19.5 Monitoring Server Performance

Each Reports Server registered with Oracle Enterprise Manager has its own main page that summarizes general information about the server's status and performance. You can monitor Reports Server performance from the following place:

- Oracle Enterprise Manager Application Server Control

19.5.1 Oracle Enterprise Manager Application Server Control

The Reports Server main page in the Application Server Control has two tabs:

- Home, shown in Figure 19–4
- Engines, shown in Figure 19–5

Figure 19–4   Home Tab of Reports Server Main Page

The **Home** tab contains the following sections:

- **General** provides the current status of the Reports Server and, if it's up, the date and time the server was started. It also includes the buttons for stopping, starting, and restarting.

- **Configuration** provides the selected Reports Server's cluster name, if it is a member of a cluster; the trace option and mode specified in its configuration file; and the maximum number of jobs that can be held at one time in the Reports Server queue (this applies to all three queues, i.e., current, scheduled, and past job queues). All of these are configurable in the Reports Server configuration file.

  See Also: For more information about the configuration file:

  - Reports Server Configuration Elements (rwserverconf.dtd)

- **Response and Load** provides current statistics for CPU and memory usage, and average response time of the Reports Server.

  - **CPU Usage (%)** displays the percentage of the host machine's CPU currently utilized by the selected Reports Server.

- **Memory Usage (%)** displays the percentage of the host machine's RAM currently utilized by the selected Reports Server. This metric does not apply to in-process servers.

  > **Note:** In-process servers run in an OC4J_BI_Forms instance, and the **CPU Usage(%)** and **Memory Usage(%)** metrics are part of the OC4J_BI_Forms instance. Thus, the metrics for an in-process server are always displayed as *Unavailable.*

- **Average Response Time** displays the average number of milliseconds taken by the Reports Server to process a request from the client.

- **Jobs** provides the number of current (currently running), finished, scheduled, and failed jobs; and the number of jobs transferred from the current server to another server in the same cluster (clustered environment). If a number greater than zero appears, you can click it to drill down to a complete listing of each job in that particular queue (e.g., all failed jobs). If the number is zero, then that queue is empty and no drill down is necessary.

- **Other Server(s) Running In This Cluster**: Refer to **Viewing and Linking to Server Cluster Members** for more information on this section.

- **Dependent Components** lists the status of all of the components (e.g., Oracle HTTP Server and Oracle Application Server Web Cache) upon which the Reports Server depends. By clicking on the component names in this list you can drill into more detailed status information for each dependent component.

- **Administration** contains links to various administrative functions:

  - **Configuration** takes you to a page where you can modify the most commonly used Reports Server or engine parameters. The Edit Configuration File button in this page enables you to edit the Reports Server configuration file directly.

  - **Edit Configuration File** takes you to a page where you can directly edit the Reports Server configuration file. This page provides a useful Check Syntax function so that you can validate your changes before committing them to the configuration file.

  - **Server Trace File** takes you to a page where you can review the latest report execution tracing information for the Reports Server or download it for further analysis.

■ **Server Log** takes you to a page where you can review the latest Reports Server log file or download it for further analysis.

> **See Also:** *Oracle Enterprise Manager online help*

For more information on the available options to search for log / trace files.

Figure 19–5   Engines Tab of Reports Server Main Page

The **Engines** tab contains the following sections:

■ **Total Engines** lists the number of engines currently up and running on the Reports Server. The table below this field lists the engines by ID with their current status.

■ **Engine ID** lists the type of engines available for processing jobs on the selected Reports Server.

■ **Engines** provides the total number of the specified engine type that is currently active on the selected Reports Server.

■ **Running** provides the total number of this type of engine that is currently running on the selected Reports Server.

■ **Engine Parameters** button to directly take you to the Configuration page where you can view/modify the Engine and Report Server parameters. This page has a link using which you can edit the Report Server Configuration file directly.

19.6 Viewing and Changing the Reports Server Configuration Files

You can modify/view the Reports Server Configuration in either of the following ways:

- Configuring Selected Parameters
- Configuring the server_name.conf File

19.6.1 Configuring Selected Parameters

To view and modify some of the most commonly used engine and server parameters through Oracle Enterprise Manager:

1. Navigate to the Reports Server main page in Application Server Control for the Reports Server you want to view.

2. Click the **Configuration** link under Administration. The parameters displayed in the Configuration screen are picked up from the Reports Server's configuration file (*server_name*.conf).

   a. **Reports Engine Parameters**:

      * `Engine ID`: Displays the valid Engine IDs for the selected server.

        Select the Engine ID from a drop down menu to edit the engine parameters for the selected engine. Only those engines that are already present in the server configuration file are displayed in the drop down list.

        Note: Click the Edit Configuration File button to add a new engine.

        ---

      * `Classpath`(Optional): The directory path to the Java class specified in the class attribute. Refer to engine for more information on the various engine properties available for the Reports Server.

      * Initial Engines`(initEngine):` The number of engines you want the Reports Server to start at initialization. Refer to engine for more information on the various engine properties available for the Reports Server.

      * Maximum Engines `(maxEngine):` The maximum number of this type of engine that can run on the server. Refer to engine for more information on the various engine properties available for the Reports Server.

* Minimum Engines (`minEngine`): The minimum number of this type of engine maintained on the server. . Refer to engine for more information on the various engine properties available for the Reports Server.

* Maximum Jobs Before Shutdown(`engLife`): Displays the ceiling of jobs that must be completed before the engine is shut down. Refer to engine for more information on the various engine properties available for the Reports Server.

* Maximum Idle Before Shutdown (`maxIdle`)(minutes): The number of minutes of allowable idle time before the engine is shut down, provided the current number of engines is higher than *minEngine*. Refer to engine for more information on the various engine properties available for the Reports Server.

* `JVM Option`(Optional): The Java Virtual Machine (JVM) options to be used by the Reports Server when it starts an engine in the JVM. Refer to engine for more information on the various engine properties available for the Reports Server.

b. **Reports Server Parameters**:

* `Cache Size(MB)`: The file size in mega bytes (MB) for the Reports Server's cache. Refer to cache for more information on the `cache` properties.

* `Cache Directory` (Optional): The location of the Reports Server cache. Refer to cache for more information on the `cache` properties.

* `Maximum Cached Files` (Optional): The ceiling on the number of files allowed in the cache. Refer to cache for more information on the `cache` properties.

* `Ignore Parameters` (Optional): Any report parameters you want to be ignored when the Reports Server constructs the cache key. (The cache key is used by the Reports Server to determine if an incoming job request matches existing output in the cache.) Refer to cache for more information on the `cache` properties.

* `Idle Timeout` (minutes) (Optional): Allowable amount of time in minutes the connection can be idle. Refer to connection for more information on the various `connection` properties.

* `Maximum Connections` (Optional): The maximum number of requests that the server can service simultaneously. Requests in excess

of the `maxConnect` value return a Java exception. Refer to connection for more information on the various connection properties.

* `Maximum Queue Size:` The maximum number of jobs that can be held in a given Reports Server job queue. Refer to queue for more information on the various queue properties.

* `Trace Options:` Refer to trace for more information on the available `trace` options. The default trace option considered when no option is specified is `trace_replace`.

c. Click OK to save the changes changes made to the engine and server parameters to the Reports Server configuration file. A Confirmation page is displayed. Click Yes to restart the server.

Note: You must restart the server for your changes to take effect.

---

d. Click Cancel to discard any changes made.

19.6.2 Configuring the server_name.conf File

To view and change a Reports Server's configuration file through Oracle Enterprise Manager:

1. Navigate to the Reports Server main page in Application Server Control for the Reports Server you want to view.

> **See Also:** Navigating to the Reports Server Page in the Application Server Control for more information.

2. Under **Administration**, click the **Edit Configuration File** link.

3. Make your changes in the display window.

4. Click the **Check Syntax** button to check your XML syntax.

Note: Clicking this button does not validate the values you enter for configuration elements. For example, if an element requires that you specify a directory path, syntax checking does not validate the accuracy of your path. It just validates the XML syntax.

---

5. Click the **OK** button to save your changes or the **Cancel** button to discard your changes. Clicking **Cancel** will also return you to the Reports Server main page.

6. If you chose **OK** in the previous step, Oracle Enterprise Manager prompts you to restart the Reports Server. Changes to the Reports Server configuration file do not take effect until you restart the Reports Server.

   Changes take effect after the next server startup or restart.

   > **See Also:** Chapter 3, "Configuring OracleAS Reports Services"
   >
   > For more information about the attributes and values in the Reports Server configuration file.

19.7 Viewing and Linking to Server Cluster Members

When you cluster Reports Servers together, it's reflected on each cluster member's page in Oracle Enterprise Manager under the **Other Servers Running in This Cluster** heading. Each listed cluster member links to the page for that member.

Table 19–5 lists and describes the information the **Other Servers …** section provides for each cluster member:

Table 19–5   Information under the Other Servers Running in this Cluster Heading

Row	Description
Server Name	Lists the names of each of the other Reports Servers that are members of the same cluster that the selected Reports Server belongs to. Click a server name to hyperlink to the cluster member's home page in OEM.
Finished Jobs	Provides the total number of finished jobs currently in the listed Reports Server's Job Queue.
Current Jobs	Provides the total number of currently running jobs in the listed Reports Server's Job Queue.
Scheduled Jobs	Provides the total number of scheduled jobs currently in the listed Reports Server's Job Queue.
Failed Jobs	Provides the total number of jobs for the listed Reports Server that were stopped before completion. This includes jobs that were user-terminated or terminated with error.
Average Response Time	Lists the average number of milliseconds it takes for the listed Reports Server to process a request from the client.

19.8 Viewing Port Numbers

In many cases, you need to quickly discover the port numbers in use by the Reports Server. The Application Server Control lets you quickly find port numbers by following these steps:

1. Launch the Application Server Control. For more information on how to launch the console, refer to your Oracle Enterprise Manager documentation.

2. The console opens on the Farm page as shown in Figure 19–1, which displays the available Oracle Application Server instances.

3. Find the Oracle Application Server instance that contains the Reports Server you want to view. Click that instance to go to its main page.

4. From the Oracle Application Server instance main page, click **Ports** to go to the Ports tab. On the Ports tab, you will find a list of all of the ports currently in use by that Oracle Application Server instance, including the ports used by the Reports Server.

Figure 19–6   Application Server Control, Ports Tab

Home J2EE Applications | **Ports** | Infrastructure

Page Refreshed May 28, 2003 11:45:39 PM PDT

The Port In Use column is empty if the port is not defined or if the component is not running. The Configure column contains an icon if you can configure the port using Enterprise Manager. Otherwise, see help topic: About Oracle9iAS Port Dependencies

Component	Type	Port In Use	Suggested Port Range	Configure
OPMN	ONS Remote	6201	6200-6299	
OPMN	ONS Request	6004	6003-6099	
OPMN	ONS Local	6101	6100-6199	
Oracle HTTP Server	The nonSSL listen port	7779	7777-7877	✎
LogLoader	LogLoader management port	44001	44000-44099	
Oracle HTTP Server	The SSL listen port	4445	4443-4543	✎
OC4J_BI_Forms	Web site port	3301	3301-3400	✎
OC4J_BI_Forms	RMI server port	3101	3101-3200	✎
OC4J_BI_Forms	JMS server port	3201	3201-3300	✎
Oracle Enterprise Manager	Console	1813	1810-1830	
Oracle Enterprise Manager	Agent	1825	1810-1830	
Reports Server	osagent port	14000	14000-14010	
Reports Server	sqlnet port for 6i client	1950	1950-1960	
Oracle Discoverer	OSAgent	16001	16001-16020	
Oracle HTTP Server	The main server port		4443-4543	✎
Oracle HTTP Server	The main server port		7777-7877	✎
home	Web site port		3000-3100	✎
home	RMI server port		3101-3200	✎
home	JMS server port		3201-3300	✎

Home J2EE Applications | **Ports** | Infrastructure

19.9 Changing Your Middle-Tier

In some cases, you might want to change the Infrastructure Services used by a middle-tier instance. For example, you may want to use a different Oracle Internet Directory or Metadata Repository than the ones you chose during installation. For instructions on how to change the Infrastructure Services used by a middle-tier instance, refer to *Oracle Application Server 10g Administrator's Guide*.

20

Tuning Oracle Reports

As your reporting requests grow in size and complexity and your user base increases, you need to consider streamlining your report's performance (or your report's execution time) as much as possible. This maximizes its reach and minimizes its delivery time. Consider the following essentials before you proceed to tune Oracle Reports:

- Performance and the trade-offs that occur when improving both perceived and measurable performance.

- Costs involved.

- Computing environment's complexity.

Investigating some of these areas can result in significant performance improvements. Some may result in minor performance improvements and others may have no affect on the actual report performance but can improve the perceived performance. Perceived performance refers to events that contribute —behind-the-scenes— to the end-result (measured in terms of the final output). See Fetching Ahead, for an example of perceived performance.

This chapter provides a number of guidelines and suggestions for good performance practices in building, implementing, and tuning individual Oracle Reports. The suggestions given are general in nature and not all suggestions might apply to all cases. However, implementing some or all of the points in a given application environment should improve the performance of report execution (real and perceived).

> **Note:** This chapter does not address Oracle Reports deployment or scalability issues. Refer to the *Oracle Application Server Reports Services Scalability* white paper on OTN, (`http://otn.oracle.com`) for more information.

This chapter contains Methodology as the main section.

20.1 Methodology

You must not look at the report in isolation, but in the context of:

- The application requirements

- The *correctness* of the underlying data model

- The environment where this report will run (for example, client/server, the Web, or before firewalls)

- The degree of user interaction required

After identifying the context, gear the tuning process towards optimizing and minimizing:

- The calls to the data source

- The amount of unnecessary formatting required for the layout

To achieve these two objectives, focus your tuning on the following distinct aspects of your report:

1. Execution time

    Determine where your report is spending a majority of its execution time. Once you have accomplished this, use one of several performance tools available: to evaluate the query, review database optimization, and examine for efficiency specific pieces of code used by the report.

    See:

    - Performance Analysis Tools and

    - Accessing the Data

    for more information on how to minimize your report's execution time.

2. Formatting and layout

Examine the formatting and layout of the report information.

See :

- Formatting the Data and
- General Layout Guidelines

for more information on how to optimize your report's formatting and layout, to yield maximum productivity.

3. Runtime parameters

Set runtime parameters to maximize performance and distribution of reports. See Bursting and Distribution, for information on how distribution maximizes your reports performance.

See:

- Calling Oracle Reports from Forms and
- Running the Report

for more information on how to set / remove appropriate runtime parameters.

The following is the best way to go about improving performance:

- Make changes incrementally.
- Isolate performance bottlenecks.
- Optimize data access.
- Streamline formatting.
- Maximize runtime efficiency.

Changes made in one area can have a performance impact in another area.

20.1.1 Performance Analysis Tools

The first step towards tuning your report is determining where your report spends most of its execution time. Does it spend a large portion of the time retrieving the data, formatting the retrieved data, or waiting for run time resources/distribution? Even if your report has the most streamlined and tuned layout possible, it may be of little consequence if most of the time is spent in retrieving data, due to inefficient SQL.

20.1.1.1 Report Trace

Setting the report tracing option produces a text file that describes the series of steps completed during the execution of the report. Tracing can be set to capture all events or just specific types of events. For example, you can trace just the SQL execution or just the layout and formatting. The trace file provides abundant information, which is useful not only for performance tuning but also for debugging reports.

Tracing can be set for various methods of report execution, such as, both `.rdf` and `.jsp` report definitions containing SQL and/or non-SQL data sources (for example, XML and Text pluggable data sources).

Generating a reports trace file

To turn tracing on, do one of the following:

- In Reports Builder:

  1. Choose **Program >Tracing**.

  2. Select the **Trace Mode**.

  3. Select appropriate **Trace Options**. The trace file now logs information for the entire Reports Builder session. Refer to Chapter 3, "Configuring OracleAS Reports Services", for more information on Trace Options.

Figure 20–1   Reports Builder
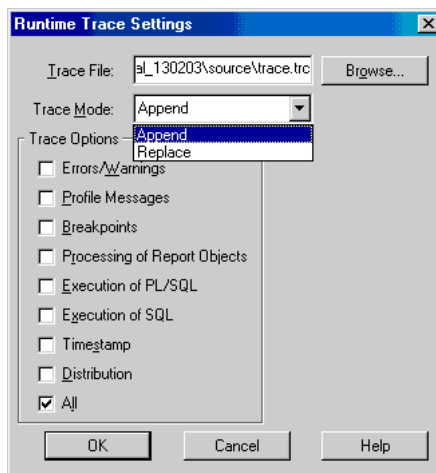
- In the `rwbuilder.conf` configuration file, specify:

  ```
  <trace traceFile="trace_file_name" traceOpts="trace_all"
  traceMode="trace_replace"/>
  ```

  > **Note:** The location of the trace file is relative to the Oracle Reports log directory (`ORACLE_HOME\reports\log`) or absolute if a full path name is specified. If you do not specify a trace file name, the default name is `hostname-rwbuilder.trc`.

- Specify tracing options via the Reports Builder built-in package (SRW) using SRW.TRACE_START, SRW.TRACE_END, SRW.TRACE_ADD_OPTION, SRW.TRACE_REM_OPTION.

- For the `rwbuilder` and `rwrun` executables, specify trace options on the command line:

  ```
  traceFile=trace_file_name traceOpts=trace_all traceMode=trace_replace
  ```

  > **Note:** The location of the trace file is relative to the current working directory or absolute if a full path name is specified. Command line tracing options override the options in the `rwbuilder.conf` file.

- For the `rwserver` executable, set the trace options in the `servername.conf` file. Separate trace files are generated for the server and the engine(s). Specify:

  ```
  <trace traceFile="trace_file_name" traceOpts="trace_all" traceMode="trace_
  replace"/>
  ```

  > **Note:** The location of the trace files is relative to the server log directory (`ORACLE_HOME\reports\log`). If you do not specify a trace file name, the default server trace file name is `serverName.trc` and the default engine trace file name is `serverName-engineName-engineNo.trc`.

- For `rwservlet`, set the trace options in the `rwservlet.properties` configuration file.

  Use a separate line for each option:

```
TRACEOPTS=TRACE_ALL

TRACEFILE=rwservlet.trc

TRACEMODE=TRACE_REPLACE
```

See Also:

- Chapter 3, "Configuring OracleAS Reports Services"

For more information on OracleAS Reports Services Trace Options.

- Appendix A, "Command Line Options"

For more information on report trace options.

Options can be combined on the command line. For example, the `TRACEOPTS=(TRACE_APP,TRACE_PRF)` means that the log file will contain information on all report objects and performance statistics.

Viewing the generated the trace file helps you determine the actual time spent in fetching data versus the time spent in formatting the report. This helps you to identify performance bottlenecks.

The following command line example generates a trace file, containing performance trace information, and replaces any previously existing trace file:

```
rwrun module=emp.rdf userid=scott/tiger@orcl destype=file desformat=pdf
desname=emp_pdf.pdf traceopts=trace_prf tracemode=trace_replace
tracefile=emp_tr.txt
```

Following is the outline of the information output to the `emp_tr.txt` trace file.

Example 20–1   Reports Builder

```
+-----------------------------------------+
| Report Builder Profiler statistics      |
+-----------------------------------------+
Total Elapsed Time: 8.00 seconds
Reports Time:    7.00 seconds (87.50% of TOTAL)
ORACLE Time:     1.00 seconds (12.50% of TOTAL)
UPI:             0.00 seconds
SQL:             1.00 seconds
TOTAL CPU Time used by process: N/A
```

Table 20–1   Reports Builder

Field	Description
Total Elapsed Time	Time spent in executing the report.
Reports Time	Time spent in formatting the retrieved data. Also displayed as a percentage of Total Elapsed Time.
ORACLE Time	Time spent in retrieving the data. Also displayed as a percentage of Total Elapsed Time.
UPI	SQL queries only. Time spent in establishing a database connection, then parsing and executing the SQL.
SQL	Time taken by the database server to fetch the data (percent of time spent executing SRW.DO_SQL() statements, EXEC_SQL statements, PL/SQL cursors, and so on.)

> **Note:**   If your data source is a non-SQL data source such as Text or an XML pluggable data source, the values for ORACLE Time, UPI, and SQL display as 0.

In Example 20–1, focus your tuning efforts on formatting (Reports Time) the data rather than on querying and fetching it.

20.1.1.2 Efficient SQL
Oracle Reports uses SQL to retrieve data from the database.

> **Note:**   Oracle Reports uses SQL for non-PDS queries only.

Inefficient SQL can cripple performance especially in large reports. Thus, anyone tuning Oracle Reports must have a good working knowledge of SQL and understand how the database executes these statements. If you are less proficient in SQL, use the Data Wizard and Query Builder in the Reports Builder. However, the wizard cannot prevent inefficient SQL from being created. For example, SQL that does not use available indexes.

To tune your report's SQL, use the Oracle database's trace functionality. SQL tracing allows you to determine the SQL statement sent to the database as well as the time taken to parse, execute, and fetch data. Once a trace file is generated, use the TKPROF database utility to generate an EXPLAIN PLAN map. The EXPLAIN

PLAN map graphically represents the execution plan used by Oracle Optimizer. For example, the Oracle Optimizer shows where full table scans have been used. This may prompt you to create an index on that table depending on the performance hit.

To turn on SQL tracing inside Reports Builder, add a report-level formula column named SQL_TRACE with the following code:

```
SRW.DO_SQL('ALTER SESSION SET SQL_TRACE=TRUE');
return(1);
```

> **Note:** You can also call SQL_TRACE using either a Before Report Form trigger, or a Before Parameter Form trigger.

The following EXPLAIN PLAN map was generated using the database's SQL trace facility. Refer to the *SQL Language Reference and Programming* documentation for more information.

Example

The statement being executed is:

```
SELECT e.ename, d.dname
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno
```

The EXPLAIN PLAN generated is:

```
OPERATION            OPTIONS      OBJECT_NAME     POSITION
------------------ ----------- --------------- --------
SELECT STATEMENT
  MERGE JOIN      OUTER                              1
    SORT                   JOIN                       1
      TABLE ACCESS FULL              DEPT           1
    SORT                   JOIN                       2
      TABLE ACCESS FULL              EMP            1
```

When you tune data for Oracle Reports, understand that the Oracle RDBMS provides two optimizers: cost-based and rule-based. By default, the cost-based optimizer constructs an optimal execution plan geared towards throughput, i.e., process all rows accessed using minimal resources. You can influence the optimizer's choice by setting the optimizer approach and goal, and gathering statistics for cost-based optimization. While the cost-based optimizer removes most of the complexity involved in tuning SQL, understanding the distribution of the

data and the optimizer rules allow you to choose the preferred method and gives you greater control over the execution plan. For example, in your SQL statement, you could do one of the following:

- Provide optimizer hints with the goal of best response time, i.e., process the first row accessed using minimal resources.

- Decide that an index is not needed.

> **Note:** For large queries, it is imperative to do one of the following:
>
> - Activate the cost-based optimizer and gather statistics by using the DBMS_STATS package, the COMPUTER STATISTICS option, or the ANALYZE command.
>
> - Optimize all SQL following the rules laid out by the rule-based optimizer.

The *Oracle Application Server* documentation provides more information on the database optimizer's functionality.

20.1.1.3 PL/SQL

Use the ORA_PROF built-in package to tune your report's PL/SQL program units. The procedures, functions, and exceptions in the ORA_PROF built-in package allow you to track the amount of time that pieces of your code takes to run.

Example

```
PROCEDURE timed_proc (test VARCHAR2) IS
i PLS_INTEGER;
BEGIN
ORA_PROF.CREATE_TIMER('loop2');
ORA_PROF.START_TIMER('loop2');
ColorBand_Program_Unit;
ORA_PROF.STOP_TIMER('loop2');
TEXT_IO.PUTF('Loop executed in %s seconds.\n',
ORA_PROF.ELAPSED_TIME('loop2'));
ORA_PROF.DESTROY_TIMER('loop2');
END;
```

This procedure creates a timer, starts it, runs a subprogram, stops the timer, and displays the time it took to run. It destroys the timer when finished.

Refer to *Reports Builder online help* for more information on the ORA_PROF built-in package.

Implement PL/SQL program units performing a significant amount of database operations as stored database procedures. Stored procedures run directly on the Oracle database and perform operations more quickly than local PL/SQL program units. Local PL/SQL program units use the Reports Builder's PL/SQL parser, then the database's SQL parser, and also include a network trip.

PL/SQL program units that do not perform any database operations should be coded as locally as possible using the *Program Units* node in the Object Navigator. Localizing the PL/ SQL program unit has a performance advantage over executing PL/SQL from an external PL/SQL library. Use external PL/SQL libraries only when the benefits of code sharing can be utilized.

The SRW.DO_SQL() built-in procedure should be used as sparingly as possible. Each call to the SRW.DO_SQL() built-in procedure necessitates parsing and binding the command and opening a new cursor like a normal query. Unlike a normal query, this operation will occur each time the object owning the SRW.DO_SQL() built-in procedure fires.

For example, a PL/SQL block in a formula column calls the SRW.DO_SQL() built-in procedure and the data model group returns 100 records. In this case, the parse/bind/ create cursor operation occurs 100 times. Therefore, use the SRW.DO_SQL() built-in procedure for operations that cannot be performed using normal SQL (for example, to create a temporary table or any other form of DDL), and in places where it will be executed sparingly (for example, in triggers that are only fired once per report).

The primary reason to use the SRW.DO_SQL() built-in procedure is to perform DDL operations, such as creating or dropping temporary tables. For example, have the SRW.DO_SQL() built-in procedure to create a table. The table's name is determined by a parameter entered in the Runtime Parameter Form.

Example

```
SRW.DO_SQL (`CREATE TABLE' || :tname ||
`(ACCOUNT NUMBER
NOT NULL PRIMARY KEY, COMP NUMBER (10,2))');
```

20.1.1.4 Java Stored Procedures

Java stored procedures enable you to implement business logic at the server level; thereby, improving application performance, scalability, and security. Oracle9i allows PL/SQL and Java stored procedures to be stored in the database. Typically,

SQL programmers who want procedural extensions favor PL/SQL and Java programmers who want easy access to Oracle data favor Java. Although Java stored procedures offer extra flexibility, there is some overhead involved. Balance the trade off between performance and flexibility based on your individual needs.

Refer to the *Oracle9i Java Stored Procedures Developer's Guide* for more information on Java stored procedures.

20.1.1.5 The Java Importer

Although Oracle PL/SQL provides a powerful and productive development environment, it is sometimes necessary to integrate with external application services and providers. As many of these external application services and providers are increasingly offering integration points in Java. Oracle Reports integrates with the Oracle Java Importer to facilitate the invocation of business logic contained in external middle-tier Java classes. The Java Importer declaratively creates a PL/SQL *wrapper* package for each class you select and exposes the methods identified in the class through PL/SQL functions and procedures. This allows you to instantiate, use, and destroy the Java object instances when the report is run. While this powerful extension insulates you from having to write Java code yourself, there is some overhead involved. Separate PL/SQL packages are generated for every class specified. The PL/SQL generator performs type translations when it generates the PL/SQL packages from the Java methods. Any time a Java object instance is created using the *new* function in the PL/SQL package and generated by the Java Importer, the result is stored in a variable of type JOBJECT. Java Object persistence must be carefully handled because accumulating large numbers of global references without removing them increases the JVM's memory consumption.

20.1.2 Accessing the Data

If your performance measuring tools show that the report spends a large amount of time accessing data from the data source(s), you need to review the structure of the data and determine how the data is being used. Inefficient schema design has a dramatic affect on the performance of a report. For example, an overly normalized data model can result in many avoidable joins or queries.

20.1.2.1 Non-SQL Data Sources

To publish data from any data source use the pluggable data source architecture in Oracle Reports. Out-of-the-box Oracle Reports supports non-SQL data sources, such as the XML, Text, and JDBC pluggable data sources. Both XML and Text pluggable

data sources can be accessed through a remote URL (even across firewalls). If speed is a concern, download the data locally and use the local data stream rather than a remote URL. You can also specify the domains for which you can bypass a proxy server.

The XML pluggable data source supports runtime XML data validation. Select the *Validate Data Source* check box in the XML query wizard to activate runtime XML data validation. Selecting *Validate Data Source* ensures that the XML data is verified as it is fetched against the data definition specified in the DTD or in the XML schema. This is a very costly operation and proves to be useful only when you develop the report and not during production. You will see a noticeable performance difference when the XML data stream is very large.

You can specify either an XML schema or a DTD schema for the data definition. An XML schema forces type checking. Whereas, a DTD schema does not require type checking as all data is treated as strings.

> **Note:** Ensure that the data types of the non-SQL sources match — columnwise.

You can also specify an extensible style sheet language (XSL) file for the XML data stream to convert it from any format into a simple row set/row data feed. It is better to have data in the correct format to start with, unless you need to apply the XSL at run time.

Pluggable Text data sources support the use of cell wrappers. This causes the file format level delimiter to be ignored for every field that has a wrapper defined. Avoid using cell wrappers unless really required.

The JDBC pluggable data source supports JDBC bridges, as well as, thick and thin JDBC drivers. Selecting the driver directly impacts the fetching of data. The choice depends on the application and the database being used. Using a native driver generally results in better performance.

20.1.2.2 Database Indexes

Columns used in a SQL WHERE clause should be indexed. The impact of indexes used on columns in the master queries of a report are minor, as these queries access the database once. To improve performance significantly, indexes should be used on any linked columns in the detail query.

> **Note:** Lack of appropriate indexes can result in many full-table scans and slows down performance.

20.1.2.3 Calculations

Within a report (either through summary or formula columns) ensure that most of the calculations are performed by the data source. In case of SQL queries, calculations are performed on the database rather than on the data retrieved by the report. User-defined functions and procedures stored by the database can also be included in the query select list of an Oracle9i or a JDBC query. This is more efficient than using a local function, since the calculated data is returned as part of the result set from the database.

Example

The following PL/SQL function can be stored in the Oracle9i database:

```
CREATE OR REPLACE FUNCTION CityState (
    p_location_id world_cities.location_id%TYPE)
    RETURN VARCHAR2 is
    v_result VARCHAR2(100);
BEGIN
    SELECT city || ','||state
        INTO v_result
        FROM world_cities
    WHERE location_id = p_location_id;
    RETURN v_result;
END CityState;
```

This function returns the city separated by a comma, a space, and the state. This formatting is done at the database level and passed back to the report to display.

In the report, the SQL query would look like:

```
SELECT location_id, citystate(location_id)"City
& State" FROM world_cities
```

The result would look like this:

```
LOCATION_ID CITY & STATE
----------- -------------------------
          1 Redwood Shores, California
          2 Seattle, Washington
          3 Los Angeles, California
          4 New York, New York
```

20.1.2.4 Redundant Data

A report's query should ideally select only required columns and not unrequired columns (redundant query) as this affects performance. The fewer queries you have, the faster your report will run. Single-query data models execute more quickly than multi-query data models. However, situations can arise where a report not only needs to produce a different format for different users, but also needs to utilize different query statements. Although this can be achieved by producing two different reports, it may be desirable to have a single report for easier maintenance. In this instance, the redundant queries should be disabled using the `SRW.SET_MAXROW()` built-in procedure.

Example

The following code used in the *Before Report* trigger will disable, either Query_Emp or Query_Dept, depending on the user parameter:

```
IF :Parameter_1 = 'A' then
    SRW.SET_MAXROW('Query_Emp',0);
ELSE
    SRW.SET_MAXROW('Query_Dept',0);
END IF;
```

> **Note:** The only meaningful place to use the `SRW.SET_MAXROW()` built-in procedure is in the *Before_Report trigger* (after the query has been parsed). Calling the `SRW.SET_MAXROW()` built-in procedure after this point raises the `SRW.MAXROW_UNSET` built-in exception. The query will still be parsed and bound, but no data will be returned to the report.

You can define a query based either on an XML or a Text pluggable data source by selecting the fields to be used in the query (i.e., all available fields or a subset). If you must use a subset of the fields, do so at the query level using parameters, as opposed to fetching all the values and filtering them using a group filter or layout level format triggers.

20.1.2.5 Break Groups

Limit the number of break groups to improve your report's performance. Oracle Reports sets the break level for each column in the data model that has the *break order* property set except the lowest child group.

For a SQL query, Oracle Reports appends this as an extra column to the ORDER BY clause in the query. The fewer columns in the ORDER BY clause, the lesser the work the database has to do before returning the data in the required order. Creating a break group may render an ORDER BY clause redundant inspite of defining it as part of the query. Remove any such ORDER BY clauses as it requires extra processing by the database.

If your report requires the use of break groups, set the *break order* property for as few columns as possible. A break order column is indicated by a small arrow to the left of the column name in the group in the *Reports Builder Data Model* view. Each break group above the lowest child group of a query requires at least one column to have the **break order** property set. Removing the break order from columns where sorting is not required increases performance.

Limit break groups to a single column whenever possible. These columns should be as small as possible and be database columns (as opposed to summary or formula columns) wherever feasible. Both conditions help the local caching that Oracle Reports does, before the data is formatted for maximum efficiency. Clearly, these conditions cannot always be met but can increase efficiency whenever utilized.

20.1.2.6 Group Filters

Group filters reduces the number of records retrieved from the data source. Filtering takes place after the query returns the data (from the data source) to Reports Builder. Even if the filter is defined to display only the top five records, the result set returned to reports will contain all the records returned by the query. Hence, it is more efficient to incorporate the group filter functionality into the query's WHERE clause or into the *maximum rows* property, whenever possible. This restricts the data returned by the database.

20.1.2.7 To Link or not to Link

There are a number of ways to create data models that include more than one table. Consider the standard case of the dept/emp join, with the requirement to create a report that lists all the employees in each department in the company. You can create either of the following:

- Single query

```
SELECT d.dname, e.ename
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno
```

- Two queries with a column link based on deptno

```
SELECT deptno, dname FROM dept
SELECT deptno, ename FROM emp
```

When you design the data model in the report, minimize the actual number of queries by using fewer large multi-table queries, rather than several simple single-table queries. Every time a query is run, Oracle Reports needs to parse, bind, and execute a cursor. A single query report returns all the required data in a single cursor, rather than many cursors. With master-detail queries, the detail query will be parsed, bound, and executed again for each master record retrieved. In this example, it is more efficient to merge the two queries and use break groups to create the master-detail effect.

Keep in mind that the larger and more complex a query gets, the more difficult it is to be maintained. You need to decide when to achieve the balance between performance and maintenance requirements.

20.1.3 Formatting the Data

After the data is retrieved from the data source, Oracle Reports generates the report layout and formats the output. The time taken for a paper layout depends on a number of factors, but generally comes down to:

- The work required to prevent an object from being overwritten by another object.
- The efficiency of any calculations or functions performed in the format triggers.

The rules for a Web layout are a little different as Oracle Reports does not:

- Own the Web page
- Control the rendering mechanism

It merely *injects* data into a regular JSP page.

20.1.3.1 Paper Layout

When generating a default paper layout, Oracle Reports wraps a frame around virtually every object to prevent the objects from being overwritten, when the report is run. At runtime, every layout object (frames, fields, boilerplate, etc.,) is examined to determine the likelihood of that object being overwritten. In some situations (for example, boilerplate text column headings) when there is clearly no risk of the objects being overwritten, the immediately surrounding frame is removed. This reduces the number of objects that Oracle Reports must format and consequently improves performance.

Extra processing is required when an object has an undefined size (variable, expanding, or contracting either horizontally and / or vertically). In this case, Oracle Reports must determine the instance of the object's size, before formatting that object and those around it. There is no processing overhead involved for objects assigned a fixed size, as the size and positional relationships between the objects is known.

The following guidelines helps to improve performance when creating a paper layout:

- Make your non-graphical layout objects (for example, boilerplate text or fields with text) fixed in size by setting the *Vertical Elasticity* and *Horizontal Elasticity* properties of the field to *Fixed*. In particular, setting the size of repeating frames and their contents to fixed, improves performance. Variable (size) non-graphical objects require more processing overhead, because Reports Builder must determine their size before formatting them. However, the overhead for fixed non-graphical objects is less, since the additional processing is not required.

- Make your graphical layout objects (for example, images and graphs) variable in size by setting the *Vertical Elasticity* and *Horizontal Elasticity* properties of the objects to *Variable*. Fixed graphical objects require more processing overhead as their contents have to be scaled to fit. Variable objects grow or shrink with the contents eliminating the need for scaling.

- Make text fields span a line (maximum) and ensure that their contents fit within the specified width (for example, use the SUBSTR function). If a text field spans more than a line, Reports Builder must use its word wrapping algorithm to format that field. Ensuring the text field takes only one line to format avoids the processing overhead of the word wrapping algorithm.

- Minimize the use of different formatting attributes (for example, fonts) within the same field or boilerplate text, because it takes longer to format.

- Use the SUBSTR function in the report query to truncate the data at the database level, instead of truncating a character string from a field in the Report Builder layout.

- For paper layout only reports, `.rdf` and `.rep` files run faster than a `.jsp` file, because the serialized formats of a `.rdf` or a `.rep` file do not require parsing. Additionally, a `.rep` file runs faster than a `.rdf` file as it is optimized for the current platform.

20.1.3.1.1 Format Triggers in a Paper Layout Format triggers can dynamically disable, enable, and change the appearance of an object. Exercise caution when using them

as they fire each time an instance of their associated object is produced and formatted (at runtime).

Consider the following example, on the use of format triggers:

A tabular report includes a single repeating frame that expands vertically and has the Page Protect property set to On. As the report is formatted, there is room for one more line at the bottom of the first page. Oracle Reports starts to format the next instance of the repeating frame and fires its associated format trigger. One of the objects inside the repeating frame is found to have expanded and this instance of the repeating frame is moved to the following page. The format trigger for the repeating frame is fired again. Although the repeating frame only appears once (at the top of the second page), the format trigger has fired twice. DML should not be performed in a format trigger, because you are not sure how many times the format trigger will fire for a particular object.

With the example above, had the format trigger contained an INSERT statement, then two rows of data would have been inserted.

Format triggers can be used against repeating frames to filter data. However, by introducing filtering at appropriate levels, you not only improve a report's performance but also reduce the complexity required for this type of a report.

Use the following filtering order whenever possible:

- Modify the SQL statement to prevent the data being returned from the server.

- Use the group filter to introduce filtering in the Data Model.

- Use **return false** inside the format trigger.

Format triggers should be placed at the highest level possible in the object/frame hierarchy so that the trigger fires at the lowest possible frequency. For example:

Figure 20–2   Format Triggers
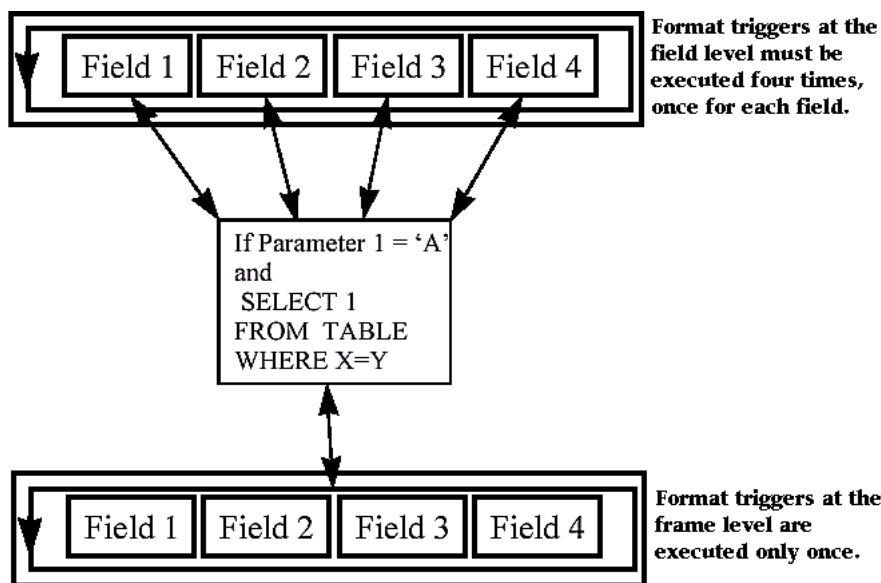
Maximize the efficiency of the code, whenever you define any triggers or PL/SQL program units within Oracle Reports. For example, to change the display attributes of a field dynamically to draw attention to values outside the norm, change the attributes using individual built-ins such as the `SRW.SET_TEXT_COLOR()` built-in procedure.

Refer to the *PL/SQL User's Guide and Reference* for general PL/SQL tuning issues.

Assigning a transparent border and fill pattern to layout objects (for example, frames and repeating frames) improves performance, as these objects are not rendered as a bitmap file.

20.1.3.2 Web Layout and JSP Report Definition

In Oracle Reports, you can use your favorite Web authoring tool to design the static portion of your Web page and then use Reports Builder to insert the dynamic portion (data) into appropriate sections of the page. A poorly designed Web page impacts perceived performance. Alternatively, you can use pre-defined Oracle9i Web templates to build the Web page.

Avoid including Java code in a JSP file (mixing business and data access Java code with presentation logic) as it increases the JSP's footprint and limits the efficient use and management of system resources.

Customized formatting of a Web page is always an expensive operation. Any type of formatting that cannot be natively achieved through Oracle Reports (e.g., change the foreground color of a data block) should be done using Java. We discourage the use of PL/SQL wrappers for formatting purposes.

A `.jsp` report definition can contain both a paper layout definition and a Web layout definition. Oracle Reports always formats the paper layout definition first when executing the report, since the Web layout section of a JSP report could contain an `<rw:include>` tag referencing a paper layout object. If your JSP report does not reference any paper layout objects at all, we recommend using the `SUPPRESSLAYOUT` command line option to prevent Oracle Reports executing the paper layout formatting.

See the *Oracle9iAS Best Practices in Performance and Scalability*: *Application Design and Configuration* white paper on OTN, (`http://otn.oracle.com`), for more performance tips on JSP technology.

20.1.4 General Layout Guidelines

20.1.4.1 Fetching Ahead

Oracle Reports enables you to display data such as total number of pages or grand totals, in the report margins or on the report header pages. This option, although useful, forces the entire report to be "fetched ahead". Fetching-ahead requires the entire report to be processed before the first page can be output. The usual model is to format pages as and when required.

Although the fetched-ahead functionality does not affect the overall time the report takes to generate, it affects the amount of temporary storage required and the time taken before the first page can be viewed. This is an example of *perceived performance* as opposed to actual performance. If the report is to be output to the screen in a production environment, fetching ahead should be avoided unless the performance variance is deemed acceptable.

20.1.4.2 Bursting and Distribution

With the introduction of report bursting, a report layout can be made up of three distinct sections: header, body, and trailer. A report can comprise all three sections, or it can be viewed as three separate reports within one report. Oracle Reports

allows you to control bursting at group record level offering a further level of granularity. This is made possible by the Distribution and Repeat On properties for each individual section. The performance gain is evident when bursting is used in conjunction with distribution, allowing each section of a report to have multiple formats and sent to multiple destinations. Once the distribution options has been set the report needs only to be run once, to be output to multiple destinations with a single execution of the query(s). Previously the report had to be executed multiple times.

When you implement bursting and distribution in a report, you can generate *section-level distribution* by setting the Repeat On property for a section to a data model break group, which generates an instance of the section for each column record of that break group. Then, you can distribute each instance of the section as appropriate (for example, to individual managers in the MANAGER group).

If you set the Repeat On property for more than one of the Header, Main, and Trailer sections of a report, all Repeat On property values must be set to the **same data model break group**. If the Repeat On property for any one of the Header, Main, and Trailer sections is set to a different data model break group, Oracle Reports raises any of the following messages:

```
REP-0069: Internal Error
REP-57054: In-Process job terminated: Terminated with error
REP-594: No report output generated.
```

20.1.5 Calling Oracle Reports from Forms

Applications built using Forms Builder and Reports Builder require reports on data that has already been retrieved/updated by the Oracle Application Server Forms Services section of the application. The tight product integration between Oracle Reports and Oracle Application Server Forms Services allows you to pass blocks of data between the associated products and removes the need for subsequent queries. This technique referred to as query partitioning ensures that Oracle Reports is responsible for formatting data and ignores dynamic alteration of queries via triggers and lexical parameters.

Passing data between Oracle Application Server Forms Services and Oracle Reports is achieved using record groups and data parameters, in conjunction with the `Run_Report_Object` built-in (for calling Oracle Reports from OracleAS Forms Services). This is the only supported way to call Oracle Reports from OracleAS Forms Services.

Refer to the *Integrating Oracle9iAS Reports in Oracle9iAS Forms* white paper on OTN, (`http://otn.oracle.com`), for more information on calling a report from an Oracle Application Server Forms Services application.

> **Note:** Unless data parameters are unreasonably large or the queries particularly complicated, the perceived performance improvements should be negligible. Additionally, only top level groups in a report can accept data parameters passed from forms.

20.1.6 Running the Report

You can further affect the overall performance by setting specific runtime options:

- Reports Builder automatically runs an error check on paper layout definitions and bind variables. Set the runtime parameter `RUNDEBUG=NO` to turn off this extra error checking at runtime.

- For JSP report definitions, Reports Builder performs tag validation and checks for items such as duplicate field identification or malformed attributes. This feature is useful only during the design phase, but not in the production environment. By default, Tag validation in OracleAS Reports Services is off. To turn this option on, use the `VALIDATETAG=YES` option in your `http` request.

- By default, the `RECURSIVE_LOAD` command line option used by both `rwrun` and `rwservlet` commands is set to yes causing invalid external references of PL/SQL program units to automatically recompile. Set the `RECURSIVE_LOAD` option to no in a production environment, because this is useful only in a development environment.

- For SQL queries, Oracle Reports takes advantage of the Oracle database's array processing capabilities for data fetching. This allows records to be fetched from the database in batches instead of one at a time, resulting in fewer calls to the database. However, array processing requires more memory on the execution platform to store the arrays of records returned. To reduce the network load (number of network trips) in a production environment, set the value of the `ARRAYSIZE` command line parameter (defined in kilobytes) to a large value.

- Set the `LONGCHUNK` parameter to as large a value as possible, if your report uses the LONG, CLOB, or BLOB data types to retrieve large amounts of data. This reduces the number of increments taken by Oracle Reports to retrieve long values. On an Oracle8*i* or Oracle9i server, use the more efficient CLOB or BLOB data types, instead of LONG or LONG RAW.

- If the paper layout Parameter Form is not required, set the `PARAMFORM` option to `NO`.

Use the `COPIES` parameter carefully when printing to PostScript. Setting `COPIES` to a value greater than 1 requires that Oracle Reports save the pages in a temporary storage, in order to collate them. This increases the amount of temporary disk space used and the overhead of writing additional files results in slow performance.

Part V

Appendices

Part V contains appendices that provide additional, detailed information about functioning in Reports Builder and in the OracleAS Reports Services environment. It includes information about Oracle Reports commands and their associated command line options, details about Reports-related environment variables, as well as how to register reports in OracleAS Portal using batch scripts.

Part V contains the following appendices:

- Appendix A, "Command Line Options"
- Appendix B, "Reports-Related Environment Variables"
- Appendix C, "Batch Registering Reports in OracleAS Portal"

A

Command Line Options

This appendix contains descriptions and examples of command line options that can be used with one or more of the following commands: `rwclient`, `rwrun`, `rwbuilder`, `rwconverter`, `rwservlet`, `rwcgi`, and `rwserver`. Each option description includes a table that indicates which commands can use which option keywords.

Note: For examples of using command line options in your runtime URL, see Running Report Requests.

The following topics are discussed in this appendix:

- Command Overview

- Command Line Syntax

- General Usage Notes

- Command Line Keywords

A.1 Command Overview

This section provides a brief description of the commands whose keywords/options are discussed in this appendix, including:

- rwclient

- rwrun

- rwbuilder

- rwconverter

- rwservlet

- rwcgi

- rwserver

Each command description includes a list of the keywords that can be used with it. In the command line, you must use the keyword along with its option. When you enter a command line, you can use the keywords in any order.

A.1.1 rwclient

The rwclient command parses and transfers a command line to the specified (or default) Reports Server. All file names and paths specified in rwclient refer to files and directories on the server machine, except for files specified in the CMDFILE keyword. In this case, the CMDFILE specified will be read and appended to the original command line (of which CMDFILE is a part) before being sent to the Reports Server. The runtime engine will not re-read the CMDFILE.

Keywords used with rwclient

The brackets surrounding each keyword in this list are there to create a separation between keywords and has no other significance.

```
rwclient [ACCESSIBLE] [ARRAYSIZE] [AUTHID] [AUTOCOMMIT] [BACKGROUND] [BCC]
[BLANKPAGES] [BUFFERS] [CACHELOB] [CC] [CELLWRAPPER] [CMDFILE][COLLATE] [COPIES]
[CUSTOMIZE] [DATEFORMATMASK] [DELIMITED_HDR] [DELIMITER] [DESFORMAT] [DESNAME]
[DESTINATION] [DESTYPE] [DISTRIBUTE] [ENGINERESPONSETIMEOUT] [EXPIRATION]
[EXPRESS_SERVER] [FROM] [JOBNAME] [JOBTYPE] [LONGCHUNK] [MODE] [MODULE|REPORT]
[NONBLOCKSQL] [NOTIFYFAILURE][NOTIFYSUCCESS][NUMBERFORMATMASK] [OLAP_CON]
[ONFAILURE] [ONSUCCESS] [ORIENTATION] [OUTPUTIMAGEFORMAT] [PAGESIZE] [PDFCOMP]
[PDFEMBED] [READONLY] [RECURSIVE_LOAD] [REPLYTO] [REPORT|MODULE] [ROLE]
[RUNDEBUG] [SCHEDULE] [SERVER][SQLTRACE][SUBJECT][SUPPRESSLAYOUT][TOLERANCE]
[TRACEFILE] [TRACEMODE] [TRACEOPTS] [USERID]
```

Examples

Example 1: Running a paper report to a browser (cache)

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb DESFORMAT=pdf
DESTYPE=cache
```

Example 2: Sending a report output to a file

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb desformat=pdf
DESTYPE=file DESNAME=c:\mydir\test
```

Example 3:   Sending a report output to a printer

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb desformat=pdf
DESTYPE=printer DESNAME=myprinter
```

Example 4:   Sending a report output via e-mail

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb desformat=pdf
DESTYPE=mail DESNAME="emp1@comp.com, emp2@comp.com" cc="emp3@comp.com"
bcc="mgr@comp.com" replyto="me@comp.com" from="me@comp.com"
```

Example 5:   Sending a report output to WebDAV (any WebDAV server or OracleAS Portal WebDAV)

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb
desformat=htmlcss DESTYPE=webdav
DESNAME="http://myusername:mypassword@mywebdavserv.com/mydir/test.html"
```

Example 6:   Sending a report output to OracleAS Portal

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb
DESTYPE=oracleportal desformat=PDF PAGEGROUP=mypagegrp OUTPUTPAGE=reports_output
ITEMTITLE=pushtoportal STATUSPAGE=result
```

Example 7:   Sending an XML PDS report output to a file

```
rwclient server=myrepserv report=myxmlpdstest.rdf destype=file desformat=PDF
desname=c:\mydir\my.pdf
```

Example 8:   Sending a JDBC PDS report output to a file

```
rwclient server=myrepserv report=myjdbcpdstest.rdf destype=file desformat=PDF
desname=c:\mydir\myxml.pdf p_jdbcpds=sybuser/sybpwd@server1.mydomain.com:1300
```

Example 9:   Distributing a report output to multiple destinations:

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb DISTRIBUTE=yes
DESTINATION=c:\mydistribute.xml
```

Example 10:   Running scheduled reports

```
rwclient server=myrepserver report=test.rdf SCHEDULE="every first fri of month
from 15:53 Oct 23, 1999 retry 3 after 1 hour" destype=file desformat=pdf
desname=test.pdf
```

Example 11:   Using a secured Reports Server

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb desformat=pdf
```

```
destype=file desname=test.pdf AUTHID=myadmin/myadmin
```

Example 12:   Running a report with e-mail notification

```
rwclient server=myrepserver report=test.rdf userid=scott/tiger@mydb destype=file
desformat=pdf desname=test.pdf NOTIFYSUCCESS="emp@comp.com"
NOTIFYFAILURE="admin@comp.com"
```

Example 13:   Running a report that specifies a URL to be fetched with the URL
engine:

```
rwclient server=myrepserver report=test.rdf userid=scott/tiger@mydb destype=file
desformat=pdf desname=test.pdf JOBTYPE=rwurl
URLPARAMETER="http://www.oracle.com"
```

Example 14:   Running a report with the traceopts enabled

```
rwclient server=myrepserver report=test.rdf userid=scott/tiger@mydb destype=file
desformat=pdf desname=test.pdf TRACEOPTS=trace_prf TRACEMODE=trace_replace
```

A.1.2 rwrun

The rwrun command runs a report using the OracleAS Reports Services in-process
server. When you run a .rep file, the PL/SQL is already compiled and will not be
recompiled. If you are running an .rdf file, the PL/SQL is automatically
recompiled, if necessary. It becomes necessary if the report wasn't compiled and
saved from Reports Builder or the platform or version on which you were running
the report is incompatible with the platform on which it was last compiled and
saved.

Keywords used with rwrun
The brackets surrounding each keyword in this list are there to create a separation
between keywords and has no other significance.

```
rwrun [ACCESSIBLE] [ARRAYSIZE] [AUTHID] [AUTOCOMMIT] [BCC] [BLANKPAGES]
[BUFFERS] [CACHELOB] [CC] [CELLWRAPPER] [CMDFILE] [COLLATE] [COPIES] [CUSTOMIZE]
[DATEFORMATMASK] [DELIMITED_HDR] [DELIMITER] [DESFORMAT] [DESNAME] [DESTINATION]
[DESTYPE] [DISTRIBUTE] [EXPRESS_SERVER] [FROM] [LONGCHUNK] [MODE]
[MODULE|REPORT] [NONBLOCKSQL] [NOTIFYFAILURE][NOTIFYSUCCESS][OLAP_CON]
[ONFAILURE] [ONSUCCESS] [ORIENTATION] [OUTPUTIMAGEFORMAT] [PAGESIZE]
[PAGESTREAM] [PDFCOMP] [PDFEMBED] [PRINTJOB] [READONLY] [RECURSIVE_LOAD]
[REPLYTO] [REPORT|MODULE] [ROLE] [RUNDEBUG] [SAVE_RDF]
[SQLTRACE][SUBJECT][SUPPRESSLAYOUT][TRACEFILE] [TRACEMODE] [TRACEOPTS] [USERID]
```

Examples

Example 1: Customizing a report

```
rwrun userid=scott/tiger@mydb report=emp.rdf CUSTOMIZE=empcustomize.xml
destype=file desformat=pdf desname=emp.pdf
```

Example 2: Running a paper report to a browser (cache)

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf destype=cache
Example 3: Sending report output to a file
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf DESTYPE=file
DESNAME=c:\mydir\test.pdf
```

Example 4: Sending a report output to a printer

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf DESTYPE=printer
DESNAME=myprinter
```

Example 5: Sending a report output via e-mail

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf DESTYPE=mail
DESNAME="emp1@comp.com, emp2@comp.com"cc="emp3@comp.com"bcc="mgr@comp.com"
replyto="me@comp.com" from="me@comp.com"
```

Example 6: Sending a report output to WebDAV (any WebDAV server or OracleAS Portal WebDAV)

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=htmlcss DESTYPE=webdav
DESNAME="http://myusername:mypassword@mywebdavserv.com/mydir/test.html"
```

Example 7: Sending a report output to OracleAS Portal

```
rwrun report=test.rdf userid=scott/tiger@mydb DESTYPE=oracleportal desformat=PDF
PAGEGROUP=mypagegrp OUTPUTPAGE=reports_output ITEMTITLE=pushtoportal
STATUSPAGE=result
```

Example 8: Sending an XML PDS report output to a file

```
rwrun report=myxmlpdstest.rdf destype=file desformat=PDF desname=c:\mydir\my.pdf
```

Example 9: Sending a JDBC PDS report output to a file

```
rwrun report=myjdbcpdstest.rdf destype=file desformat=PDF
desname=c:\mydir\myxml.pdf p_jdbcpds=sybuser/sybpwd@server1.mydomain.com:1300
```

Example 10: Distributing a report output to multiple destinations

```
rwrun report=test.rdf userid=scott/tiger@mydb DISTRIBUTE=yes
DESTINATION=c:\mydistribute.xml
```

Example 11:   Using a secured Reports Server

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf destype=file
desname=test.pdf AUTHID=myadmin/myadmin
```

Example 12:   Running a report with e-mail notification

```
rwrun report=test.rdf userid=scott/tiger@mydb destype=file desformat=pdf
desname=test.pdf NOTIFYSUCCESS="emp@comp.com" NOTIFYFAILURE="admin@comp.com"
```

Example 13:   Running a report with the traceopts enabled

```
rwrun report=test.rdf userid=scott/tiger@mydb destype=file desformat=pdf
desname=test.pdf TRACEOPTS=trace_prf TRACEMODE=trace_replace
```

A.1.3 rwbuilder

The rwbuilder command invokes Reports Builder. When you include a
REPORT|MODULE keyword with the rwbuilder command at the command
prompt, then press Enter, Reports Builder opens with the specified report
highlighted in the Object Navigator. When no report is specified, Reports Builder
opens with a Welcome dialog offering you the choice of opening an existing report
or creating a new one.

Keywords used with rwbuilder

The brackets surrounding each keyword in this list are there to create a separation
between keywords and has no other significance.

```
rwbuilder [ACCESSIBLE] [ARRAYSIZE] [AUTOCOMMIT] [BLANKPAGES] [BUFFERS]
[CACHELOB] [CMDFILE] [EXPRESS_SERVER] [LONGCHUNK] [MODULE|REPORT] [NONBLOCKSQL]
[OLAP_CON] [ONFAILURE] [ONSUCCESS] [OUTPUTIMAGEFORMAT] [PAGESIZE] [PRINTJOB]
[READONLY] [REPORT|MODULE] [RUNDEBUG] [SAVE_RDF][SQLTRACE] [SUPPRESSLAYOUT]
[TRACEFILE] [TRACEMODE] [TRACEOPTS] [USERID] [WEBSERVER_DEBUG] [WEBSERVER_
DOCROOT] [WEBSERVER_PORT]
```

Example: rwbuilder report=myrep.rdf userid=scott/tiger@mydb

A.1.4 rwconverter

The rwconverter command enables you to convert one or more report definitions or PL/SQL libraries from one storage format to another. For example, you can use rwconverter to:

- Combine a report file with an XML file to create a new report

- Convert a report stored in an .rdf file to a .rep, .rex, .jsp, or .tdf (template) file.

> **Note:** When a report is converted to a template, only objects in the report's header and trailer sections and margin area are used in the template. Objects in the main section are ignored.

- Convert a report stored in a .rex file to an .rdf or a template (.tdf file)

- Convert a library stored in the database to a .pld or .pll file

- Convert a library stored in a .pld file into a database library or a .pll file

- Convert a library stored in a .pll file into a database library of a .pld file

> **Note:** When you convert a report that has an attached library, convert the .pll files attached to the report before converting the .rdf/.rex file.

- Batch create a PL/SQL script that batch registers reports in OracleAS Portal

In some cases, rwconverter will automatically compile the report's PL/SQL as part of the conversion process. Provided your conversion destination is not a .rex file, rwconverter automatically compiles PL/SQL under the following conditions:

- Converting to a .rep file. If there are compile errors, rwconverter displays an error message and the .rep file is not created.

- Using a .rex file as the source

- Using a report created on another platform as the source

In all other situations, you must compile the report's PL/SQL yourself (e.g., using **Program >> Compile >> All** in Reports Builder).

> **Note:** Fonts are mapped when a report is opened by Reports Builder or Reports Runtime, not during the conversion.

Keywords used with rwconverter

The brackets surrounding each keyword in this list are there to create a separation between keywords and has no other significance.

```
rwconverter [BATCH] [CMDFILE] [CUSTOMIZE] [DEST] [DTYPE] [DUNIT] [FORMSIZE]
[NOTIFYFAILURE] [OVERWRITE] [PAGESIZE] [SOURCE] [STYPE] [USERID] [P_OWNER] [P_
SERVERS] [P_NAME] [P_DESCRIPTION] [P_PRIVILEGE] [P_AVAILABILITY] [P_TYPES] [P_
FORMATS] [P_PRINTERS] [P_PFORMTEMPLATE] [P_TRIGGER] [RECURSIVE_LOAD][UPGRADE_
PLSQL]
```

Example: rwconverter scott/tiger@mydb stype=rdffile source=inven1.rdf
dtype=xmlfile dest=inven1_new.xml

A.1.5 rwservlet

`rwservlet` translates and delivers information between either a Web server or a J2EE Container (for example, OC4J) and the Reports Server, enabling you to run a report dynamically from your Web browser. Optionally, it can use the In-Process Server, which reduces the maintenance and administration of the Reports Server by providing a means for starting the server automatically, whenever it receives the first request from the client.

> **Note:** When you use the `rwservlet` command to run a JSP, you can use all keywords applicable to `rwservlet`. For more information on running a JSP with `rwservlet`, see Running Report Requests.

Keywords used with rwservlet

The brackets surrounding each keyword in this list are there to create a separation between keywords and has no other significance.

```
rwservlet [ACCESSIBLE] [ARRAYSIZE] [AUTHID] [AUTOCOMMIT] [BACKGROUND][BCC]
[BLANKPAGES] [BUFFERS] [CACHELOB] [CC] [CELLWRAPPER] [CMDKEY] [CONTENTAREA*]
[COLLATE] [COPIES] [CUSTOMIZE] [DATEFORMATMASK] [DELAUTH] [DELIMITED_HDR]
[DELIMITER] [DESFORMAT] [DESNAME] [DESTINATION] [DESTYPE] [DISTRIBUTE]
[ENGINERESPONSETIMEOUT] [EXPIRATION] [EXPIREDAYS] [EXPRESS_SERVER] [FROM]
[GETJOBID] [GETSERVERINFO] [HELP] [ITEMTITLE] [JOBNAME] [JOBTYPE] [KILLJOBID]
```

[LONGCHUNK] [MIMETYPE] [MODE] [MODULE|REPORT] [NONBLOCKSQL] [NOTIFYFAILURE]
[NOTIFYSUCCESS] [OLAP_CON] [ONFAILURE] [ONSUCCESS] [ORIENTATION] [OUTPUTFOLDER*]
[OUTPUTIMAGEFORMAT] [OUTPUTPAGE] [PAGEGROUP] [PAGESIZE] [PAGESTREAM] [PARAMFORM]
[PARSEQUERY] [PDFCOMP] [PDFEMBED] [READONLY] [RECURSIVE_LOAD] [REPLACEITEM]
[REPLYTO] [REPORT|MODULE] [ROLE] [RUNDEBUG] [SCHEDULE] [SERVER] [SHOWAUTH]
[SHOWENV] [SHOWMAP][SHOWJOBID] [SHOWJOBS] [SHOWMYJOBS]
[SITENAME*][SQLTRACE][STATUSFORMAT] [STATUSFOLDER*] [STATUSPAGE] [SSOCONN]
[SUBJECT][SUPPRESSLAYOUT][TOLERANCE] [TRACEMODE] [TRACEOPTS] [URLPARAMETER]
[USERID]

* maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or
Oracle WebDB Release 2.2.

> **Note:** The following keywords are commands rather than
> keyword-value pairs, i.e., commands are entered by themselves
> without a corresponding value: SHOWENV, SHOWJOBS, SHOWMAP,
> SHOWMYJOBS, KILLJOBID, PARSEQUERY, DELAUTH, GETJOBID,
> and GETSERVERINFO.

Examples

http://myias[1].mycomp.com:*7779*[2]/reports/rwservlet?server=myrepserver+report=myrep
o.rdf+desname=sample.pdf+desformat=pdf+destype=file

http://myias.mycomp.com:*7779*/reports/rwservlet?server=myrepserver+authid[3]=myrepu
ser/myreppassword+report=myrepo.rdf+desname=sample.pdf+desformat=pdf+destype=fil
e

Example 1:   Running a paper report to a browser (cache)

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=pdf+**DESTYPE=cache**

Example 2:   Sending a report output to a file

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=pdf+**DESTYPE=file+DESNAME=c:\mydir\test**.

Example 3:   Sending a report output to a printer

---

[1] Oracle Application Server instance

[2] Port where the Reports servlet is running

[3] Secured server only

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=pdf+DESTYPE=printer+DESNAME=myprinter
```

Example 4: Sending a report output via e-mail

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=pdf+DESTYPE=mail+DESNAME="emp1@comp.com,emp2@c
omp.com"+CC="emp3@comp.com"+BCC="mgr@comp.com"+REPLYTO="me@comp.com"+FROM=me@com
p.com"
```

Example 5: Sending a report output to WebDAV (any WebDAV server or OracleAS Portal WebDAV)

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=htmlcss+DESTYPE=webdav+DESNAME="http://myusern
ame:mypassword@mywebdavserv.com/mydir/test.html"
```

Example 6: Sending a report output to OracleAS Portal

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+destype=oracleportal+desformat=PDF+PAGEGROUP=mypagegrp+O
UTPUTPAGE=reports_output+ITEMTITLE=pushtoportal+STATUSPAGE=result
```

Example 7: Sending an XML PDS report output to a file

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=myxmlpdst
est.rdf+destype=file+desformat=PDF+DESNAME=c:\mydir\my.pdf
```

Example 8: Sending a JDBC PDS report output to a file

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=myjdbcpds
test.rdf+destype=file+desformat=PDF+desname=c:\mydir\myxml.pdf+p_
jdbcpds=sybuser/sybpwd@server1.mydomain.com:1300
```

Example 9: Distributing a report output to multiple destinations

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+DISTRIBUTE=yes+DESTINATION=c:\mydistribute.xml
```

Example 10: Running scheduled reports

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserver+report=test.rd
f+SCHEDULE="every first fri of month from 15:53 Oct 23, 1999 retry 3 after 1
hour"+destype=file+desformat=pdf+desname=test.pdf
```

Example 11: Using a secured Reports Server

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=pdf+destype=file+desname=test.pdf+AUTHID=myadm
in/myadmin
```

Example 12: Using a key file

```
http://myias.mycomp.com:7779/report/rwservlet?key1
```

where

key1=key defined in the cgicmd.dat file (the keyname should be the first parameter)

or

```
http://myias.mycomp.com:7779/report/rwservlet?server=myrepserv+userparam=12+CMDK
EY*=keyname
```
*when used with cmdkey, it can be anywhere in the URL

Example 13: Running a report with a Parameter Form

```
http://myias.mycomp.com:7779/rwservlet?server=myrepserver+report=test.rdf+userid
=scott/tiger@mydb+destype=cache+desformat=htmlcss+PARAMFORM=yes
```

Example 14: Running a report with e-mail notification

```
http://myias.mycomp.com:7779/rwservlet?server=myrepserver+report=test.rdf+userid
=scott/tiger@mydb+destype=file+desformat=pdf+desname=test.pdf+NOTIFYSUCCESS="emp
@comp.com"+NOTIFYFAILURE="admin@comp"
```

Example 15: Running a report that specifies a URL to be fetched with the URL engine

```
http://myias.mycomp.com:7779/rwservlet?server=myrepserver+report=test.rdf+userid
=scott/tiger@mydb+destype=file+desformat=pdf+desname=test.pdf+JOBTYPE=rwurl+URLP
ARAMETER="http://www.oracle.com"
```

Example 16: Running a report with the traceopts enabled

```
http://myias.mycomp.com:7779/rwservlet?server=myrepserver+report=test.rdf+userid
=scott/tiger@mydb+destype=file+desformat=pdf+desname=test.pdf+TRACEOPTS=trace_
prf+TRACEMODE=trace_replace
```

A.1.6 rwcgi

Like rwservlet, rwcgi (the Common Gateway Interface (CGI) translates and delivers information between either a Web server or a J2EE Container (for example,

OC4J) and the Reports Server, enabling you to run a report dynamically from your Web browser.

> **Note:** With Oracle Reports 10*g*, `rwcgi` is deprecated (maintained only for backward compatibility); instead, use JSPs or servlets (`rwservlet`).

Keywords used with rwcgi

The brackets surrounding each keyword in this list are there to create a separation between keywords and has no other significance.

```
rwcgi [ACCESSIBLE] [ARRAYSIZE] [AUTHID] [AUTOCOMMIT] [BACKGROUND][BCC]
[BLANKPAGES] [BUFFERS] [CACHELOB] [CC] [CELLWRAPPER] [COLLATE] [CONTENTAREA*]
[COPIES] [CUSTOMIZE] [DATEFORMATMASK] [DELAUTH][DELIMITED_HDR] [DELIMITER]
[DESFORMAT] [DESNAME] [DESTINATION] [DESTYPE] [DISTRIBUTE]
[ENGINERESPONSETIMEOUT] [EXPIRATION] [EXPIREDAYS] [EXPRESS_SERVER]
[FROM][GETJOBID][ITEMTITLE] [JOBNAME] [JOBTYPE][KILLJOBID][LONGCHUNK] [MIMETYPE]
[MODE] [MODULE|REPORT] [NONBLOCKSQL] [NOTIFYFAILURE] [NOTIFYSUCCESS] [OLAP_CON]
[ONFAILURE] [ONSUCCESS] [ORIENTATION] [OUTPUTFOLDER*] [OUTPUTIMAGEFORMAT]
[OUTPUTPAGE] [PAGEGROUP] [PAGESIZE] [PAGESTREAM]
[PARAMFORM][PARSEQUERY][PDFCOMP][PDFEMBED] [READONLY] [RECURSIVE_LOAD]
[REPLACEITEM] [REPLYTO] [REPORT|MODULE] [ROLE][RUNDEBUG][SCHEDULE][SERVER]
[SHOWENV][SHOWMAP][SHOWJOBID][SHOWJOBS] [SHOWMYJOBS] [SHOWAUTH] [SITENAME*]
[SQLTRACE] [STATUSFOLDER*] [STATUSPAGE] [SUBJECT][SUPPRESSLAYOUT][TOLERANCE]
[TRACEMODE] [TRACEOPTS] [USERID]
```

* maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

Examples

```
http://mywebserver.com⁴:7779⁵/cgi-bin/rwcgi.exe⁶?server=myrepserver+report=myrepo
.rdf+desname=sample.pdf+desformat=pdf+destype=file
```

```
http://mywebserver.com:7779/cgi-bin/rwcgi.sh⁷?server=myrepserver+authid⁸=myrepus
er/myreppassword+report=myrepo.rdf+desname=sample.pdf+desformat=pdf+destype=file
```

---

[4] Web server running the CGI scripts

[5] Web server listener port

[6] Web server installed on the Windows operating system

[7] Web server installed on any UNIX operating system

[8] Secured server only

Example 1:   Running a paper report to a browser (cache)

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+DESFORMAT=pdf+DESTYPE=cache
```

Example 2:   Sending a report output to a file

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+desformat=pdf+DESTYPE=file+DESNAME=c:\mydir\test
```

Example 3:   Sending a report output to a printer

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+desformat=pdf+DESTYPE=printer+DESNAME=myprinter
```

Example 4:   Sending a report output via e-mail

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+desformat=pdf+DESTYPE=mail+DESNAME="emp1@comp.com,
emp2@comp.com+CC=emp3@comp.com"+BCC="mgr@comp.com"+REPLYTO="me@comp.com+from=me@
comp.com"
```

Example 5:   Sending report output to WebDAV (any WebDAV server or OracleAS Portal WebDAV)

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+desformat=htmlcss+DESTYPE=webdav+DESNAME="http://myuserna
me:mypassword@mywebdavserv.com/mydir/test.html"
```

Example 6:   Sending a report output to OracleAS Portal

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+DESTYPE=oracleportal+desformat=PDF+PAGEGROUP=mypagegrp+OU
TPUTPAGE=reports_output+ITEMTITLE=pushtoportal+STATUSPAGE=result
```

Example 7:   Sending XML PDS report output to a file

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=myxmlpdste
st.rdf+DESTYPE=file+desformat=PDF+desname=c:\mydir\my.pdf
```

Example 8:   Sending a JDBC PDS report output to a file

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=myjdbcpdst
est.rdf+destype=file+desformat=PDF+desname=c:\mydir\myxml.pdf+p_
jdbcpds=sybuser/sybpwd@server1.mydomain.com:1300
```

Example 9:   Distributing report output to multiple destinations

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+DISTRIBUTE=yes+DESTINATION=c:\mydistribute.xml
```

Example 10: Running scheduled reports

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserver+report=test.rdf
+SCHEDULE="every first fri of month from 15:53 Oct 23, 1999 retry 3 after 1
hour"+destype=file+desformat=pdf+desname=test.pdf
```

Example 11: Using a secured Reports Server

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+report=test.rdf+u
serid=scott/tiger@mydb+desformat=pdf+destype=file+desname=test.pdf+AUTHID=myadmi
n/myadmin
```

Example 12: Using a key file

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?key1
```
where key1=key defined in the cgicmd.dat file (the keyname should be the first
parameter)

or

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserv+userparam=12+CMDK
EY=key1
```
*when used with cmdkey, it can be anywhere in the URL

Example 13: Running a report with a Parameter Form

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserver+report=test.rdf
+userid=scott/tiger@mydb+destype=cache+desformat=htmlcss+PARAMFORM=yes
```

Example 14: Running a report with e-mail notification

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserver+report=test.rdf
+userid=scott/tiger@mydb+destype=file+desformat=pdf+desname=test.pdf+NOTIFYSUCCE
SS="emp@comp.com"+NOTIFYFAILURE="admin@comp.com"
```

Example 15: Running a report that specifies a URL to be fetched with the URL
engine

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserver+report=test.rdf
+userid=scott/tiger@mydb+destype=file+desformat=pdf+desname=test.pdf+JOBTYPE=rwu
rl+URLPARAMETER="http://www.oracle.com"
```

Example 16: Running a report with the traceopts enabled

```
http://mywebserver.com:7779/cgi-bin/rwcgi.exe?server=myrepserver+report=test.rdf
```

```
+userid=scott/tiger@mydb+destype=file+desformat=pdf+desname=test.pdf+TRACEOPTS=t
race_prf+TRACEMODE=trace_replace
```

A.1.7 rwserver

The rwserver command invokes the Reports Server. The Reports Server processes client requests, which includes ushering them through its various services, such as authentication and authorization checking, scheduling, caching, and distribution (including distribution to custom or pluggable output destinations). Reports Server also spawns runtime engines for generating requested reports, fetches completed reports from the Reports cache, and notifies the client that the job is ready.

Keywords used with rwserver

The brackets surrounding each keyword in this list are there to create a separation between keywords and has no other significance.

```
rwserver[AUTHID] [AUTOSTART] [BATCH] [SERVER] [SHUTDOWN] [TRACEOPTS] [INSTALL]
[UNINSTALL][JVMOPTIONS]
```

Example: rwserver -install myrepserver batch=yes autostart=yes

A.2 Command Line Syntax

Following is the syntax for a command line, where keyword=value is a valid command line option:

```
rwclient REPORT|MODULE=runfile USERID=username/password@database
[ [keyword=]value|(value1, value2, ...) ] SERVER=server_name
```

Keywords must be specified and can be used in any order following the command.

A.3 General Usage Notes

- All file names and paths specified in the client command line refer to files and directories on the server machine, except for any file specified for the command file keyword (CMDFILE=). In this case, the CMDFILE specified will be read and appended to the original command line (of which CMDFILE is a part) before being sent to the Reports Server. The runtime engine will not re-read the CMDFILE.

- If you don't specify a path for a keyword value that includes a file name, The Reports Server will try to find the file from the REPORTS_PATH environment variable.

A.4 Command Line Keywords

A.4.1 ACCESSIBLE

Table A–1 indicates which commands can use the ACCESSIBLE keyword.

Table A–1   Commands that can use ACCESSIBLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use ACCESSIBLE to specify whether accessibility-related features offered through Oracle Reports are enabled (YES) or disabled (NO) for PDF output.

Syntax   ACCESSIBLE={YES|NO}

Values

- YES   Accessibility features are enabled for PDF output.

- NO   Accessibility feature is not enabled for PDF output.

Default   NO

A.4.2 ARRAYSIZE

Table A–2 indicates which commands can use the ARRAYSIZE keyword.

Table A–2   Commands that can use ARRAYSIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use ARRAYSIZE to specify the size (in kilobytes) for use with Oracle9*i* array processing. Generally, the larger the array size, the faster the report will run.

Syntax   ARRAYSIZE=*n*

Values

n   A number from 1 through 9999 (no comma is used with thousands). This means that Reports Runtime can use this number of kilobytes of memory per query in your report.

Default   10

Usage Notes   ARRAYSIZE can be used when running JSP-based Web reports from the command line.

A.4.3 AUTHID

Table A–3 indicates which commands can use the AUTHID keyword.

Table A–3   Commands that can use AUTHID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	yes

Description   Use AUTHID to specify the user name and the password to be used to authenticate users to the restricted Reports Server. User authentication ensures that the users making report requests have access privileges to run the requested report.

Syntax   AUTHID=*username*/*password*

Values

username/password Any valid user name and password created in OracleAS Portal. See your DBA to create new users accounts in OracleAS Portal.

Default   None

Usage Notes

- AUTHID can be used when running JSP-based Web reports from the command line.

- If you have a Single Sign-On environment, then the Oracle Application Server Single Sign-On Server will perform the authentication step and pass only the user name to the Reports Server in AUTHID.

A.4.4 AUTOCOMMIT

Table A–4 indicates which commands can use the AUTOCOMMIT keyword.

Table A–4   Commands that can use AUTOCOMMIT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use AUTOCOMMIT to specify 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 Data changes are committed to the database automatically.
- NO   Data changes are not committed to the database until the COMMIT command runs or one of the PL/SQL commands that cause the data to be committed runs.

Default   NO

Usage Notes   AUTOCOMMIT can be used when running JSP-based Web reports from the command line.

A.4.5 AUTOSTART

Table A–5 indicates which commands can use the AUTOSTART keyword.

Table A–5   Commands that can use AUTOSTART

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	no	no	yes

Description   Use AUTOSTART to specify that the Reports Server will automatically start after initial installation and after a reboot, without requiring a user logon.

Syntax   AUTOSTART={YES|NO}

Values

- YES The service will automatically start at reboot, without requiring a user logon.

- NO   The service will not automatically start at reboot, requiring a user logon.

Default NO

Usage Notes The AUTOSTART keyword is only recognized on Microsoft Windows platforms.

A.4.6 BACKGROUND

Table A–6 indicates which commands can use the BACKGROUND keyword.

Table A–6   Commands that can use BACKGROUND

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description BACKGROUND specifies whether a report on the server should be run synchronously (NO) or asynchronously (YES).

> **Note:** The BACKGROUND system parameter is deprecated in Oracle Reports. BACKGROUND is used only on the command line.

Syntax BACKGROUND={YES|NO}

Values

- YES Runs the report asynchronously. The client sends the call to the server, then continues with other processes without waiting for the report job to complete. If the client process is killed, the job is canceled.

- NO Runs the report synchronously. The client waits for the report to queue, be assigned to a runtime engine, run, and finish.

Default NO

Usage Notes If BACKGROUND=YES is used with rwbuilder or rwrun, a warning is issued and the keyword is ignored.

A.4.7 BATCH

Table A–7 indicates which commands can use the BATCH keyword.

Table A–7   Commands that can use BATCH

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	yes

Description   For all relevant commands, the BATCH option tells the server to run in no-UI mode. How it is used across commands is similar in that no UI is produced by the application when running from a command line that includes BATCH=YES. For example, for rwserver this allows the server to be run from scripts and remote agents so that no server dialog comes up while it is running.

With rwconverter, BATCH suppresses all terminal input and output in order to convert reports/libraries without user intervention. With rwserver, BATCH turns the server window dialog off (YES) or on (NO) to display or suppress process messages.

Syntax   BATCH={YES|NO}

Values

- YES Suppresses all terminal input and output (report is run in the background). This is the default for rwrun.

- NO Allows special terminal input and output. For rwconverter, the **Convert** dialog box is displayed, and when you accept the dialog box, the conversion is performed.

Default   NO

Usage Notes
- If BATCH=YES, error messages are sent to SYSOUT. For more information on SYSOUT, see DESNAME.

- If BATCH=YES, PARAMFORM=YES is invalid because it is not meaningful to have the Runtime Parameter Form appear in batch mode.

A.4.8 BCC

Table A–8 indicates which commands can use the BCC keyword.

Table A–8   Commands that can use BCC

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use BCC to specify e-mail recipient(s) of a blind courtesy copy (that is, one in which the names of specified recipients are not visible (published) to other recipients).

> **Note:**   A blind copy is one in which the names of specified recipients are not visible (published) to other recipients.

Syntax   BCC=emailid | ("emailid","emailid", ...)

Values

emailid A valid e-mail address in the form *someone*@*foo*.com.

Default   None

Usage Notes

- To specify more than one e-mail address, enclose the list of addresses in quotation marks and separate each address in the list with a comma.

- Related keywords include BCC, CC, FROM, REPLYTO, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

- BCC can be used when running JSP-based Web reports from the command line.

A.4.9 BLANKPAGES

Table A–9 indicates which commands can use the BLANKPAGES keyword.

Table A–9   Commands that can use BLANKPAGES

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use BLANKPAGES to specify 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 Print all blank pages.

- NO Do not print blank pages.

Default   YES

Usage Notes   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.

A.4.10 BUFFERS

Table A–10 indicates which commands can use the BUFFERS keyword.

Table A–10   Commands that can use BUFFERS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use BUFFERS to specify 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

n A number from 1 through 9999 (note that thousands are not expressed with any internal punctuation, e.g., a comma or a decimal point). For some operating systems, the upper limit might be lower.

Default   640

Usage Notes

- If this setting is changed in the middle of your session, then the change does not take effect until the next time the report is run.

- BUFFERS can be used when running JSP-based Web reports from the command line.

A.4.11 CACHELOB

Table A–11 indicates which commands can use the CACHELOB keyword.

Table A–11   Commands that can use CACHELOB

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use CACHELOB to specify whether to cache retrieved ORACLE large object or objects in the temporary file directory on the Reports Server (specified in the environment variable REPORTS_TMP or by the tempDir property of the engine element in the Reports Server configuration file, *server_name*.conf; note that a tempDir setting overrides a REPORTS_TMP setting.).

Syntax   CACHELOB={YES|NO}

Values

- YES The LOB will be cached in the temporary file directory.

- NO The LOB will not be cached in the temporary file directory.

Default   YES

Usage Notes

- You can only set this option on the command line.

- If the location of the temporary file directory on the server 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 database multiple times.

- CACHELOB can be used when running JSP-based Web reports from the command line.

A.4.12 CC

Table A–12 indicates which commands can use the CC keyword.

Table A–12 Commands that can use CC

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use CC to specify e-mail recipient(s) of a courtesy copy.

Syntax   CC=emailid | ("emailid","emailid", ...)

Values

emailid A valid e-mail address in the form *someone@foo*.com.

Default   None

Usage Notes

- To specify more than one e-mail address, enclose the list of addresses in quotation marks and separate each address in the list with a comma.

- Related keywords include BCC, CC, FROM, REPLYTO, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.4.13 CELLWRAPPER

Table A–13 indicates which commands can use the CELLWRAPPER keyword.

Table A–13 Commands that can use CELLWRAPPER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use CELLWRAPPER to specify the character or characters that displays around the delimited cells in your report output.

Syntax   CELLWRAPPER=*value*

Values

value Any alphanumeric character or string of alphanumeric characters.

Table A–14   Valid Values - General

Value	Description
"	A double quotation mark displays on each side of the cell
'	A single quotation mark displays on each side of the cell

Table A–15   Valid Values - Reserved

Value	Description
tab	A tab displays on each side of the cell
space	A single space displays on each side of the cell
return	A new line displays on each side of the cell
none	No cell wrapper is used

Table A–16   Valid Values - Escape Sequences based on the ASCII Character set

Value	Description
\t	A tab displays on each side of the cell
\n	A new line displays on each side of the cell

Default   None

Usage Notes

- This keyword can only be used if you have specified DESFORMAT=DELIMITED or DESFORMAT=DELIMITEDDATA.

- The cell wrapper is different from the actual delimiter. The cell wrapper specifies what character appears around delimited data. The delimiter indicates the boundary or break point between two pieces of data.

A.4.14 CMDFILE

Table A–17 indicates which commands can use the CMDFILE keyword.

Table A–17   Commands that can use CMDFILE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	yes	no	no	no

Description Use CMDFILE to call a file that contains one report's command line options. The file called must be an ASCII file, either .txt or any other ASCII-type file.

CMDFILE differs from the cgicmd.dat file, in that CMDFILE can contain one command line for one report, where the cgicmd.dat file can contain multiple key-identified commands for multiple reports. Additionally, the CMDFILE keyword can be used along with other arguments in a command line; while, when you use the key argument associated with cgicmd.dat, it is the only argument that appears in the command line.

The CMDFILE keyword enables you to run a report without specifying a large number of options each time you invoke a run command.

Syntax CMDFILE=*filename*

Values

filename Any valid command file name.

Default None

Usage Notes

- With rwservlet and rwcgi, use the CMDKEY keyword to refer to a key in the cgicmd.dat file in lieu of using the CMDFILE keyword.

- A command file can reference another command file.

- The syntax for a command line you specify in the command file is identical to that used on the command line.

- Values entered on the command line override values specified in command files. For example, suppose you specify rwclient from the command line with COPIES set to 1 and CMDFILE set to RUNONE (a command file). The RUNONE file also specifies a value for COPIES, but it is set to 2. The value specified for COPIES in the command line (1) overrides the value specified for COPIES in the RUNONE file (2). Only one copy of the report will be generated.

- The value for this keyword might be operating system-specific.

A.4.15 CMDKEY

Table A–18 indicates which commands can use the CMDKEY keyword.

Table A–18   Commands that can use CMDKEY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	no	no

Description   Use CMDKEY to call a key-identified command line in the `cgicmd.dat` file. For example:

```
http://yourwebserver/reports/rwservlet?cmdkey=key& …
```

Syntax   CMDKEY=*key*

Values

key The name of any key associated with a command line specified in the `cgicmd.dat` file.

Default   None

Usage Notes

- When you use CMDKEY with rwservlet, you can use it in any order in the command line (or the URL, following the question mark). With rwservlet, you can use additional command line keywords along with CMDKEY.

- CMDKEY can be used when running JSP-based Web reports from the command line.

A.4.16 COLLATE

Table A–19 indicates which commands can use the COLLATE keyword.

Table A–19   Commands that can use COLLATE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description: Use Collate to control the collating behavior when a report is output to a printer.

For example, printing three copies of a three page document with COLLATE set to YES would result in output similar to the following:

1 2 3 | 1 2 3 | 1 2 3

The order specified is the page numbers being printed. This behavior is similar to selecting the **Collate** check box in the Print dialog box.

Printing three copies of a three page document with COLLATE set to NO would result in output similar to the following:

1 1 1| 2 2 2| 3 3 3

Syntax   COLLATE={YES|NO}

Values

- YES Collates the pages when output to a printer.

- NO Does not collate the pages when output to a printer.

Default YES

A.4.17 CONTENTAREA

Table A–20 indicates which commands can use the CONTENTAREA keyword.

Table A–20   Commands that can use CONTENTAREA

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use CONTENTAREA to specify the Oracle9iAS Portal Release 1 content area to which report output should be pushed. This keyword is maintained for backward compatibility with Oracle9iAS Portal Release 1; for backward compatibility with Oracle WebDB Release 2.2, see SITENAME. Beginning with OracleAS Portal Release 2, use PAGEGROUP.

Syntax   CONTENTAREA=*name*

Values

name The name (*internal name*) of any valid Oracle9iAS Portal Release 1 content area.

Default   None

Usage Notes

- Use of this keyword is *required* to push Oracle Reports output to Oracle9iAS Portal Release 1.

- The CONTENTAREA name should be the internal name and *not* the display name. The internal name is used to uniquely identify the Oracle9iAS Portal Release 1 component instance.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.18 COPIES

Table A–21 indicates which commands can use the COPIES keyword.

Table A–21   Commands that can use COPIES

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use COPIES to specify the number of copies of the report output to print.

Syntax   COPIES=*n*

Values

n Any valid integer from 1 through 9999 (note that thousands are not expressed with any internal punctuation, e.g., a comma or a decimal point).

Default   Taken from the Initial Value property of the COPIES parameter (the Initial Value was defined in Reports Builder at design time).

Usage Notes

- This keyword is ignored if DESTYPE is not PRINTER.
- If COPIES is left blank on the **Runtime Parameter Form**, then it defaults to one.

A.4.19 CUSTOMIZE

Table A–22 indicates which commands can use the CUSTOMIZE keyword.

Table A–22   Commands that can use CUSTOMIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	yes	yes	yes	no

Description   Use CUSTOMIZE to specify an Oracle Reports XML file to be run against the current report. The XML file contains customizations (for example, changes to the layout or data model) that change the report definition in some way.

Syntax   CUSTOMIZE=*filename*.xml | (*filename1*.xml, *filename2*.xml, …)

Values

filenamen.xml The names of the files that contain a valid XML report definition, with path information prefixed to the name(s) if necessary. (if the files are not located in a path specified in the REPORTS_PATH registry or SourceDir property for the engine element).

> **Note:**   For more information on customizing reports at runtime with XML customization files, see Customizing Reports with XML.

Default   None

Usage Notes

- Typically, the file extension of an XML report definition is .xml, but it does not have to be when it is used with the CUSTOMIZE keyword.

- CUSTOMIZE can be used when running JSP-based Web reports from the command line.

A.4.20 DATEFORMATMASK

Table A–23 indicates which commands can use the DATEFORMATMASK keyword.

Table A–23   Commands that can use DATEFORMATMASK

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use DATEFORMATMASK to specify how date values display in your delimited report output.

Syntax   DATEFORMATMASK=*mask*

Values

mask Any valid date format mask.

Default   None

Usage Notes

- This keyword can only be used if you have specified DESFORMAT=DELIMITED or DESFORMAT=DELIMITEDDATA.

    > **Note:**   For valid DATEFORMATMASK values see the *Reports Builder online help* topic, "Date and Time Format Mask Syntax."

- DATEFORMATMASK can be used when running JSP-based Web reports from the command line.

A.4.21 DELAUTH

Table A–24 indicates which commands can use the DELAUTH keyword.

Table A–24   Commands that can use DELAUTH

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use DELAUTH to delete rwservlet or rwcgi user ID cookies.

Syntax
http://*yourwebserver*/reports/rwservlet/delauth[?][server=*servername*][&authid=*username*/*password*]

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- Related keywords are SERVER and AUTHID.

A.4.22 DELIMITED_HDR

Table A–25 indicates which commands can use the DELIMITED_HDR keyword.

Table A–25   Commands that can use DELIMITED_HDR

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use DELIMITED_HDR to turn off boilerplate text (such as the report header) when running a report with DESFORMAT=DELIMITED or DESFORMAT=DELIMITEDDATA.

Syntax   DELIMITED_HDR={YES|NO}

Values

- YES Turn off all boilerplate text in the delimited output file.

- NO Leave boilerplate text as is in the delimited output file.

Default   YES

Usage Notes   This keyword can be used only if you have specified DESFORMAT=DELIMITED or DESFORMAT=DELIMITEDDATA.

A.4.23 DELIMITER

Table A–26 indicates which commands can use the DELIMITER keyword.

Table A–26   Commands that can use DELIMITER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use DELIMITER to specify the character or characters to use to separate the cells in your report output.

Syntax   DELIMITER=*value*

Values

value Any alphanumeric character or string of alphanumeric characters, such as:

Table A–27   Valid Values - General

Values	Description
,	A comma separates each cell
.	A period separates each cell

Any of these reserved values:

Table A–28   Valid Values - Reserved

Values	Description
tab	A tab separates each cell
space	A space separates each cell
return	A new line separates each cell
none	No delimiter is used

Table A–29   Valid Values - Escape Sequence based on the ASCII Character set

Values	Description
\t	A tab separates each cell
\n	A new line separates each cell

Default   Tab

Usage Notes   This keyword can be used only if you have specified
DESFORMAT=DELIMITED or DESFORMAT=DELIMITEDDATA.

A.4.24 DESFORMAT

Table A–30 indicates which commands can use the DESFORMAT keyword.

Table A–30   Commands that can use DESFORMAT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use DESFORMAT to specify the format for the job output. Use
DESFORMAT to specify the printer driver to be used when DESTYPE is FILE and
DESNAME=*filename*.

Syntax   DESFORMAT=*desformat*

Values   Any valid destination format not to exceed 1 kilobyte in length. Examples
of valid values for this keyword are listed and described in Table A–31.

Table A–31   Valid values for DESFORMAT

Value	Description
DELIMITED	The report output is sent to a file that can be read by standard spreadsheet utilities, such as Microsoft Excel. If you do not specify a delimiter (via the DELIMITER keyword), the default delimiter is a TAB. See Usage Notes.
DELIMITEDDATA	Provides the same functionality as DELIMITED, and is used when you have problems running large volume reports with DESFORMAT=DELIMITED. See Usage Notes.
HTML	This report output is sent to a file that is in HTML format.
HTMLCSS	This report output is sent to a file that includes style sheet extensions.
PDF	This report output is sent to a file that is in PDF format and can be read by a PDF viewer, such as Adobe Acrobat. PDF output is based upon the currently configured printer for your system. The drivers for the currently selected printer are used to produce the output; you must have a printer configured for the machine on which you are running the report.
POSTSCRIPT	This report output is sent to a file that is in PostScript format.

Table A–31   Valid values for DESFORMAT

Value	Description
PRINTER DEFINITION	The printer definition to use when formatting the report when DESTYPE=FILE and DESNAME=*filename*:
	If MODE=BITMAP, this is the name of the printer.
	If MODE=CHARACTER, this is the character-mode printer definition file (.prt file), such as hpl, hplwide, dec, decwide, decland, dec180, dflt, or wide. Ask your System Administrator for a list of valid *printer definition*s.
RTF	The report output is sent to a file that can be read by word processors (such as Microsoft Word). When you open the file in Microsoft Word, you must choose **View > Page Layout** to view all the graphics and objects in your report.
XML	This report output is saved as an XML file. 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 (the Initial Value was defined in Reports Builder at design time). When you run a report via Reports Builder and DESFORMAT is blank or dflt, then the current printer driver (specified in **File > Printer**) is used. If a Printer Name has not been selected using the drop down list, then Reports Builder defaults to PostScript.

Usage Notes

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

- When DESFORMAT=HTML or DESFORMAT=HTMLCSS, spaces are replaced with . This default behavior eliminates alignment issues for number values that are right-aligned. If you do not want spaces replaced with   in your HTML and HTMLCSS output, then you must set REPORTS_NO_HTML_SPACE_ REPLACE to YES. This removes the functionality of the DELIMITER command line keyword for HTML and HTMLCSS output (DELIMITER is still valid when DESFORMAT=DELIMITED).

- DESFORMAT=DELIMITED is not supported in a .dst file. In this case, Oracle Reports displays error:

```
REP-34305: Invalid keyword setting for the destid='DEST1'
```

- When `DESFORMAT=DELIMITEDDATA`, the DelimitedData driver runs off the data model and operates in much the same way as the XML driver. Since the driver runs off the data model, any formatting changes defined in the layout are not reflected in the DelimitedData output. You can set the following column properties to alter column names and exclude columns from the DelimitedData output file:

  - The XML Tag property can be used to enter a column alias.

  - The Exclude from XML Output property can be used to exclude the column from the DelimitedData output.

  The DelimitedData functionality also honors the `DELIMITER`, `CELLWRAPPER`, `NUMBERFORMATMASK`, and `DATEFORMATMASK` command line options just as the `DELIMITED` does.

- When you open RTF output generated by Oracle Reports in Microsoft Word 95 for Japanese, you may encounter anomalies in the output, such as dashes not appearing correctly. These issues are specific to Microsoft Word 95 and do not occur in Microsoft Word 97 for Japanese.

A.4.25 DESNAME

Table A–32 indicates which commands can use the `DESNAME` keyword.

Table A–32   Commands that can use DESNAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use `DESNAME` to specify the name of the cache, file, printer, OracleAS Portal, or e-mail ID (or distribution list) to which the report output will be sent.

Syntax   `DESNAME=desname`

Values

desname Any valid cache destination, file name, printer name, e-mail ID, or OraclePortal, 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 (the Initial Value was defined in Reports Builder at design time). If DESTYPE = FILE and DESNAME is an empty string, then it defaults to reportname.lis at runtime.

Usage Notes

- The value(s) for this keyword might be case sensitive, depending on your operating system.

- To send the report output by e-mail, specify the e-mail ID as you do in your e-mail application (any SMTP-compliant application). You can specify multiple user names by separating them with commas, and without spaces. For example:

  name,name,name

- In some cases, this keyword may be overridden by your operating system.

A.4.26 DEST

Table A–33 indicates which commands can use the DEST keyword.

Table A–33   Commands that can use DEST

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description Use DEST to specify the name(s) of the converted reports or libraries.

Syntax DEST={*dest_name*|(*dest_name1, dest_name2, …*)|*pathname*}

Values

dest_name Any valid report/library name or filename, or a list of valid report/library names of filenames enclosed in parentheses and separated by commas (e.g., (qanda,text,dmast)).

Default If the DEST keyword is not specified, rwconverter uses the following default names:

- If DTYPE is PLDFILE, then the DEST default name is source.pld.

- If DTYPE is PLLFILE, then the DEST default name is source.pll.

- If DTYPE is RDFFILE, then the DEST default name is source.rdf.

- If DTYPE is REPFILE, then the DEST default name is `source.rep`.

- If DTYPE is REXFILE, then the DEST default name is `source.rex`.

- If DTYPE is XMLFILE, then the DEST default name is `source.xml`.

- If DTYPE is REGISTER, then the DEST default name is the name of the SQL*Plus script output file (e.g., `output.sql`).

Usage Notes

- A list of report/library names of filenames must be enclosed in parentheses with commas separating each entry. For example:

  `(qanda,test,dmast) or (qanda, test, dmast)`

- If you have more destination names than there are source names, the extra destination names are ignored. If you have fewer destination names than there are source names, default names will be used after the destination names run out.

- The value(s) for the DEST keyword may be operating system-specific.

- When DTYPE = REGISTER, multiple destinations are not required. If you list more than one SQL*Plus script file name for DEST, only the first one is recognized. The others are ignored.

A.4.27 DESTINATION

Table A–34 indicates which commands can use the DESTINATION keyword.

Table A–34   Commands that can use DESTINATION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use the DESTINATION keyword to specify the name of an XML file that defines the distribution for the current run of the report.

> **Note:**   XML based distribution files *must* have the `.xml` extension.

Syntax   DESTINATION=*filename*.xml

Values

filename.xml The name of an XML file that defines a report or report section distribution.

Default None

Usage Notes To enable the DESTINATION keyword, you must specify DISTRIBUTE=YES on the command line. If both these keywords are specified, DESTYPE, DESNAME, and DESFORMAT are ignored if they are also specified.

> **Note:** For more information about creating advanced distributions, see Chapter 15, "Creating Advanced Distributions".

A.4.28 DESTYPE

Table A–35 indicates which commands can use the DESTYPE keyword.

Table A–35   Commands that can use DESTYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description Use DESTYPE to specify the type of device that will receive the report output. If you have created your own pluggable destination via the Oracle Reports Destination API, this is how the destination you created gets called.

Syntax
```
DESTYPE={CACHE|LOCALFILE|FILE|PRINTER|SYSOUT|MAIL|ORACLEPORTAL
|FTP|WEBDAV|name_of_pluggable_destination}
```

Values Table A–36 lists and describes the valid values for the DESTYPE keyword.

Table A–36   Valid values for the DESTYPE keyword

Value	Description
CACHE	Sends the output directly to the Reports Server cache.
LOCALFILE	Valid only for rwclient, rwcgi, and rwservlet. Sends the output to a file on the client machine, synchronously or asynchronously.
FILE	Sends the output to the file on the server named in DESNAME.

Table A–36   Valid values for the DESTYPE keyword

Value	Description
PRINTER	Sends the output to the printer on the server named in DESNAME. You must have a printer that the OracleAS Reports Services can recognize installed and running.
SYSOUT	Valid only for rwcgi. Sends the output to the client machine's default output device and forces a synchronous call.
MAIL	Sends the output to the mail users specified in DESNAME. You can send mail to any mail system that works with SMTP. **Note:** The configuration file rwbuilder.conf must include the pluginParam mailServer with the outgoing mail server name. This applies in both Windows and Solaris environment
ORACLEPORTAL	Valid for rwservlet, rwcgi, and rwclient. Sends the output to OracleAS Portal. Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SCHEDULE, SITENAME*, STATUSFOLDER*, STATUSPAGE. * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.
FTP[1]	Valid for rwservlet, rwclient, rwcgi, and rwrun. Sends the output to the specified FTP server.
WEBDAV[2]	Valid for rwservlet, rwclient, rwcgi, and rwrun. Sends the output to the specified WebDAV URL so that the report can be published directly.
name_of_pluggable_ destination	If you have created your own pluggable destination via the Oracle Reports Destination API, this is what you use to call the destination you created.

[1] For more information about this pluggable destination, see the Oracle Reports Plugin Exchange on OTN (http://otn.oracle.com/products/reports/pluginxchange/content.html)

[2] For more information about this pluggable destination, see the Oracle Reports Plugin Exchange on OTN (http://otn.oracle.com/products/reports/pluginxchange/content.html)

Default   Taken from the Initial Value property of the DESTYPE parameter (the Initial Value was defined in Reports Builder at design time).

Usage Notes

- With Oracle Reports, DESTYPE values of SCREEN and PREVIEW are no longer valid because the Reports Runtime (rwrun) user interface is obsolete. In Reports Builder (rwbuilder), you can still set the DESTYPE system parameter

to SCREEN or PREVIEW to format a report to display screen fonts or printer (preview) fonts in the Reports Builder user interface.

■ DESTYPE=printer: On Windows, the hardware-based left margin is ignored, by default. The printing origin starts from the top left corner (0,0) of the physical paper and not the printable area. This is to facilitate the design of printer hardware-based margin independent reports. Printing reports without hardware-based left margins on Windows You must ensure that your report's layout contains enough margin spacing such that your data falls within the printable area. Margin fields in the Page Setup dialog have been disabled to ensure consistency with OracleAS Reports Services. To revert to the old behavior of including the hardware margin, set the REPORTS_ADD_HWMARGIN environment variable to YES.

■ DESTYPE=ORACLEPORTAL:

The ORACLEPORTAL destination cannot be used with distribution. Instead, you can use DESTYPE=WEBDAV for advanced XML based distribution to OracleAS Portal.

Note: Ensure that the OracleAS Portal instance should be WebDAV-enabled. Refer to the *OracleAS Portal online help* for more information on how to enable WebDAV.

For more information on how to use WebDAV for distribution to OracleAS Portal, refer to the Tech note at:

http://webiv.oraclecorp.com/cgi-bin/webiv/do.pl/Get?WwwID=note:241821.

The DESTYPE=ORACLEPORTAL destination cannot be used with the rwrun executable as it causes Reports Server to stop responding. Use this destination only with rwservlet, rwclient, or rwcgi.

Before you push Oracle Reports output to Oracle9iAS Portal Release 1 ensure that you have created the following:

■ A valid OUTPUTPAGE containing at least one *item* region

■ A valid PAGEGROUP containing at least one *item* region

Additionally, you need to edit the Reports Server configuration file as follows:

1. Uncomment the destype=oraclePortal element.

```
<destination
   destype="oraclePortal"
   class="oracle.reports.server.DesOraclePortal">
   <!--property name="portalUserid"
             value="%PORTAL_DB_USERNAME%/%PORTAL_DB_
                      PASSWORD%@%PORTAL_DB_TNSNAME%"
             confidential="yes"
             encrypted="no"/-->
</destination>
```

> **Note:** In Oracle9*i* Reports 9.0.2, by default, the `portalUserid` property is uncommented and the connection string in the property points to the infrastructure database. In order to be able to push your reports output to Oracle9iAS Portal Release 1 the `portalUserid` must remain uncommented.
>
> In Oracle Reports 10*g* (9.0.4), by default, the `portalUserid` is commented out. Reports Server will determine the connection string and push the report to Oracle9iAS Portal Release 1. You need to uncomment this only if you are using a different OracleAS Portal instance.

2. Substitute the values in the `portalUserid` property with your OracleAS Portal connection information if you do not want to push an Oracle Reports output to the default Oracle9iAS Portal Release 1 instance.

> **Note:** If you do not substitute the values or uncomment the `destype` entry, you will get the following error:
>
> ```
> REP-56092: No class defined for destination type
> oracleportal
> ```

Running the request is very similar to any other out-of-the-box destinations. For example:

```
http://your_server:port/reports/rwservlet?
report=test.rdf&userid=scott/tiger@repportal&authid=pushportal/trial&destype
=oracleportal&desformat=PDF&pagegroup=PORTAL_REPORTS&outputpage=reports_
output&itemtitle=pushtoportal&statuspage=result
```

- `DESTYPE=FTP`: Running the request is very similar to any other out-of-the-box destinations. You need to specify the complete FTP URL location along with the file name. If the FTP server needs an authentication, that also needs to be part of the URL as shown in the following example:

  `http://`*your_*
  server:port`/reports/rwservlet?report=rep.jsp&`**destype=FTP&desname=ftp://user:**
  pwd@ftpServer/dir/myreport.pdf&desformat=pdf
  In the above example, the `DESTYPE` is `FTP` and the `DESNAME` value is a complete FTP URL location along with the report name `myreport.pdf`.

  To use a proxy server, edit the `destination` element (configured for FTP out-of-the-box) in the Reports Server configuration file:

  ```
  <destination destype="ftp"
  class="oracle.reports.plugin.destination.ftp.DesFTP">
  <!--property name="proxy" value="proxyinfo.xml"/-->
  </destination>
  ```

  To specify the proxy information, edit the `proxyinfo.xml` file available in the default location (*ORACLE_HOME*`\reports\conf`). Uncomment the previous property entry and specify the complete path to the `proxyinfo.xml` file as the value.

  For example, if your *ORACLE_HOME* is located in `D:\9iAS`, then the default location for `proxyinfo.xml` can be specified as:

  ```
  <destination destype="ftp"
  class="oracle.reports.plugin.destination.ftp.DesFTP">
      <property name="proxy" value="D:\\9iAS\\reports\\conf\\proxyinfo.xml"/>
  </destination>
  ```

- `DESTYPE=WEBDAV`: Running the request is very similar to any other out-of-the-box destinations. You need to specify the complete WEBDAV URL location along with the file name. If the WEBDAV server needs an authentication, that also needs to be part of the URL as shown in the following example:

  `http://`*your_*
  server:port`/reports/rwservlet?report=rep.jsp&`**destype=WebDAV&desname=http://u**
  ser:pwd@webdavServer/myreport.pdf&desformat=pdf

  In the above example, the `DESTYPE` is `WEBDAV` and the `DESNAME` value is a complete WEBDAV URL location along with the report name `myreport.pdf`.

To use a proxy server, edit the `destination` element (configured for WEBDAV out-of-the-box) in the Reports Server configuration file:

```
<destination destype="WebDav"
class="oracle.reports.plugin.destination.webdav.DesWebDAV">
     <!--property name="proxy" value="proxyinfo.xml"/-->
 </destination>
```

To specify the proxy information, edit the `proxyinfo.xml` file available in the default location (*ORACLE_HOME*\reports\conf). Uncomment the property entry above and specify the complete path to the `proxyinfo.xml` file as the value.

For example, if your *ORACLE_HOME* is located in `D:\9iAS`, then the default location for `proxyinfo.xml` can be specified as:

```
<destination destype="WebDav"
class="oracle.reports.plugin.destination.webdav.DesWebDAV">
    <property name="proxy" value="D:\\9iAS\\reports\\conf\\proxyinfo.xml"/>
</destination>
```

A.4.29 DISTRIBUTE

Table A–37 indicates which commands can use the DISTRIBUTE keyword.

Table A–37   Commands that can use DISTRIBUTE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use DISTRIBUTE to enable or disable distributing the report output to multiple destinations, as specified by the distribution list defined in the report distribution definition (defined in Reports Builder at design time) or a distribution XML file.

Syntax   DISTRIBUTE={YES|NO}

Values

- YES Distribute the report to the distribution list.

- NO Ignore the distribution list and output the report as specified by the DESNAME, DESTYPE, 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 Notes The DISTRIBUTE keyword works in close association with the DESTINATION keyword. DISTRIBUTE must have a value of YES for the DESTINATION keyword to take effect. If both these keywords are specified, DESTYPE, DESNAME, and DESFORMAT are ignored if they are also specified.

> **Note:** For more information about creating advanced distributions, see Creating Advanced Distributions.

A.4.30 DTYPE

Table A–38 indicates which commands can use the DTYPE keyword.

Table A–38   Commands that can use DTYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description Use DTYPE to specify the format to which to convert the reports or libraries.

Syntax
```
DTYPE={PLDFILE|PLLFILE|RDFFILE|REPFILE|REXFILE|TDFFILE|XMLFILE
|JSPFILE|REGISTER}
```

Values The following values apply:

- PLDFILE The converted PL/SQL libraries will be stored in files in ASCII format.

- PLLFILE The converted PL/SQL libraries will be stored in files containing source code and P-code (compiled PL/SQL).

- RDFFILE The converted report(s) will be stored in one or more report definition files (files with the .rdf extension).

- REPFILE The converted report(s) will be stored in one or more binary runfiles (files with the .rep extension).

- REXFILE The converted report(s) will be stored in one or more text files (files with the `.rex` extension).

- TDFFILE The report will be converted to a template file (files with the `.tdf` extension).

- XMLFILE The converted report(s) will be stored in an XML file (files with the `.xml` extension).

- JSPFILE The converted report(s) will be stored in a JSP file (files with the `.jsp` extension).

- REGISTER A script file is created to load each report specified by SOURCE into OracleAS Portal with the RWWWVREG.REGISTER_REPORT function. Each load function is populated with the necessary information to register the report in OracleAS Portal. By running the resulting script file in SQL*Plus against the Oracle Application Server DB Provider, you can batch register multiple reports in OracleAS Portal.

Default REPFILE

Usage Notes

- When you try to create a `.rep` file using rwconverter, the source report's PL/SQL is automatically compiled. If there are compile errors, an error message is displayed and the `.rep` file is not created. To avoid this problem, make sure you compile the source report's PL/SQL using **File > Administration > Compile Report** in Reports Builder, before you try to create a `.rep` file.

- When converting a report to a template, only objects in the report's header and trailer sections and the margin area are used in the template. Objects in the main section are ignored.

A.4.31 DUNIT

Table A–39 indicates which commands can use the DUNIT keyword.

Table A–39 Commands that can use DUNIT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description Use DUNIT to specify the destination unit of measurement to which the report should be converted. If specified, DUNIT must differ from the SOURCE

report's unit of measurement. If unspecified, the SOURCE report's unit of measurement is used.

Syntax DUNIT={CENTIMETER│CHARACTER│INCH│POINT}

Values

- CENTIMETER The converted reports will initially use centimeters as the unit of measurement

- CHARACTER The converted reports will initially use characters as the unit of measurement.

- INCH The converted reports will initially use inches as the unit of measurement.

- POINT The converted reports will initially use points as the unit of measurement

Default Null (the report's unit of measurement is used).

A.4.32 ENGINERESPONSETIMEOUT

Table A–40 indicates which command can use the ENGINERESPONSETIMEOUT keyword.

Table A–40   Commands that can use ENGINERESPONSETIMEOUT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description   Use ENGINERESPONSETIMEOUT to specify the maximum amount of time (in minutes) for an engine to update the status of the job while running a report in your environment. If it takes longer than this amount of time to update the job status for some reason (e.g., due to the engine hanging or a long blocking SQL query), then Reports Server terminates the job.

This parameter overrides the engineResponseTimeOut attribute of the engine element in the Reports Server configuration file. Refer to engine for further information about the engine element.

Syntax ENGINERESPONSETIMEOUT=*number*

Values

number A number of minutes (e.g., 5).

Default None

A.4.33 EXPIRATION

Table A–41 indicates which command can use the EXPIRATION keyword.

Table A–41   Commands that can use EXPIRATION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description Use EXPIRATION to define how long report output can exist in cache before it is deleted.

See Reusing Report Output from Cache for more information on duplicate job detection.

Syntax EXPIRATION=*time_string*

Values

time_string Is in one of two formats:

- *n*{*unit*}, for a number with an optional unit. The unit can be minute(s), hour(s), or day(s). The default unit is minute(s) if no unit is specified.

- {*Mon DD*, *YYYY*} *hh*:*mi*:*ss* am|pm {*timezone*}, for a date/time format. Date information is optional. If it isn't specified, *today* is assumed. Time zone is also optional. If it isn't specified, the Reports Server's timezone is used. The date/time is always in a US locale. This format is the same as defined in the Java DateFormat.MEDIUM type.

Default None

A.4.34 EXPIREDAYS

Table A–42 indicates which commands can use the EXPIREDAYS keyword.

Table A–42   Commands that can use EXPIREDAYS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use EXPIREDAYS to specify the number of days after which the Oracle Reports output pushed to Oracle9iAS Portal Release 1 should be expired.

Syntax   EXPIREDAYS={PERMANENT|1 day|2 days|3 days|7 days|14 days|31 days|60 days|90 days|120 days}

Values

- PERMANENT Does not expire.

- *n* days Expires after *n* days.

Default   None

Usage Notes

- Use of this keyword is *optional* to push Oracle Reports output to Oracle9iAS Portal Release 1.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.35 EXPRESS_SERVER

Table A–43 indicates which commands can use the EXPRESS_SERVER keyword.

Table A–43   Commands that can use EXPRESS_SERVER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use EXPRESS_SERVER to specify the Express Server to which you want to connect.

Syntax `EXPRESS_`
`SERVER="server=`*server*`/domain=`*domain*`/user=`*userid*`/password=`*passw*
d`"`

Syntax with RAM `EXPRESS_`
`SERVER="server=`*server*`/domain=`*domain*`/user=`*userid*`/password=`*passw*
d`/ramuser=`*ramuserid*`/rampassword=`*rampasswd*`/ramexpressid=`*ramexpi*
d`/ramserverscript=`*ramsscript*`/rammasterdb=`*ramdb*`/ramconnecttype=`
ramconn`"`

Values A valid connect string enclosed in double quotes (") where:

Table A–44   Connect String Values

Values	Description
`server`	The Express Server string (for example, `ncacn_ip_` `tcp:olap2-pc/sl=x/st=x/ct=x/sv=x/`). See below for more details on the server string.
`domain`	The Express Server domain.
`user`	The user ID to log on to the Express Server.
`password`	The password for the user ID.
`ramuser`	The user ID to log into the RDBMS.
`rampassword`	The password for the RDBMS.
`ramexpressid`	The Oracle Sales Analyzer database user ID. This is required for Oracle Sales Analyzer databases only.
`ramserverscript`	The complete file name (including the full path) of the remote database configuration file (RDC) on the server. This file specifies information such as the location of code and data databases. Using UNC (Universal Naming Convention) syntax allows multiple users to use the same connection to access the data without having to map the same drive letter to that location. UNC syntax is `\\`*ServerName*`\`*ShareName*`\` followed by any subfolders or files.
`rammasterdb`	The name of the Relational Access Manager database to attach initially. You must specify only the database file name. This database must reside in a directory that is included in the path list in ServerDBPath for Express Server. You can check the ServerDBPath in the File I/O tab of the Express Configuration Manager dialog box.

Table A–44   Connect String Values

Values	Description
ramconnecttype	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. Table A–45 describes the parameters and their settings:

Table A–45   Settings for parameters used with EXPRESS_SERVER's server value

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

Default   None

Usage Notes

- You can have spaces in the string if necessary (for example, if the user ID is John Smith) because the entire string is enclosed in 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. They are not treated specially because the entire string is enclosed in double quotes.

A.4.36 FORMSIZE

Table A–46 indicates which commands can use the FORMSIZE keyword.

Table A–46 Commands that can use FORMSIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use FORMSIZE to specify the size of the Runtime Parameter Form for the converted report in terms of the destination unit of measurement (specified using DUNIT).

Syntax   FORMSIZE=*width x height*

Values

width/height Any valid values in the specified unit of measurement.

Default   None

Usage Notes

- For non-character DUNITs, you can use a decimal to specify fractions (e.g., 8.5 x 11).

- For more information on the Runtime Parameter Form, see the PARAMFORM keyword.

A.4.37 FROM

Table A–47 indicates which commands can use the FROM keyword.

Table A–47   Commands that can use FROM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use FROM to specify the e-mail address of the sender of an e-mail.

Syntax   FROM=*emailid*

Values

emailid Any valid e-mail address in the form *someone@foo*.com.

Default   *loginid@machine_name*

Usage Notes   Related keywords include BCC, CC, FROM, REPLYTO, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.4.38 GETJOBID

Table A–48 indicates which commands can use the GETJOBID keyword.

Table A–48   Commands that can use GETJOBID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use GETJOBID to retrieve the result output of the Reports Server job with job ID **n**.

Syntax   http://*yourwebserver*/reports/rwservlet/getjobidn[?][server=*server_name*][&authid=*username*/*password*]

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- Job must be successfully finished and present in the Reports Server cache. Use SHOWJOBS to see the current list of jobs.

- Related keywords are SERVER and AUTHID.

A.4.39 GETSERVERINFO

Table A–49 indicates which commands can use the GETSERVERINFO keyword.

Table A–49   Commands that can use GETSERVERINFO

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	no	no

Description   Use GETSERVERINFO to display Reports Server information. You can choose the format (HTML or XML) in which the information is returned via &statusformat.

Syntax   http://*yourwebserver*/reports/rwservlet/getserverinfo[?][server=*server_name*][&authid=*username/password*][&statusformat={html|xml}]

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- Related keywords are SERVER and AUTHID.

A.4.40 HELP

Table A–50 indicates which commands can use the HELP keyword.

Table A–50   Commands that can use HELP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use HELP to display a help topic that lists the keywords you can use with the rwservlet command. For example:

```
http://yourwebserver/reports/rwservlet/help?command=keyword
```

Syntax `http://yourwebserver/reports/rwservlet/help`

or

`http://yourwebserver/reports/rwservlet/help?command=keyword`

Values See Syntax.

Default None

A.4.41 INSTALL

Table A–51 indicates which commands can use the INSTALL keyword

Table A–51 Commands that can use INSTALL

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	no	no	yes

Description Use INSTALL to configure an instance of the Reports Server on Microsoft Windows as a service. This keyword does not work on UNIX platforms.

Syntax `INSTALL reports_server_name`

Values

`reports_server_name` A valid name for the Reports Server instance.

Default None

Usage Notes

- If you use the AUTOSTART keyword with INSTALL, the Reports Server service will be started automatically after installation and whenever the system is restarted.

- If you use BATCH=YES with INSTALL, then none of the prompts and dialogs that normally display during installation will appear.

A.4.42 ITEMTITLE

Table A–52 indicates which commands can use the ITEMTITLE keyword.

Table A–52   Commands that can use ITEMTITLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use ITEMTITLE to specify the display name Oracle9iAS Portal Release 1 should use for Oracle Reports output. The name will display in Oracle9iAS Portal Release 1 and link to Oracle Reports output.

Syntax   ITEMTITLE=*title*

Values

title Any text. Put quotation marks around the value if the value has any character spaces in it or you are specifying the option in the cgicmd.dat file.

Default   The report file name

Usage Notes

- Use of this keyword is *optional* to push

- Oracle Reports output to Oracle9iAS Portal Release 1.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

    * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.43 JOBNAME

Table A–53 indicates which commands can use the JOBNAME keyword.

Table A–53   Commands that can use JOBNAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description   Use JOBNAME to specify the name for a job to appear in Reports Queue Manager. It is treated as a comment and has nothing to do with running the

job. If `JOBNAME` is not specified, then Reports Queue Manager shows the report name as the job name.

Syntax   `JOBNAME=string`

Values   Any job name.

Default   None

Usage Notes   `JOBNAME` can be used when running JSP-based Web reports from the command line.

A.4.44 JOBTYPE

Table A–54 indicates which commands can use the `JOBTYPE` keyword.

Table A–54   Commands that can use JOBTYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description   Use `JOBTYPE` to specify the type of job to be processed by the server. You can enter any type of job, as long as the Reports Server has an engine to process it.

Syntax   `JOBTYPE=jobtype`

Values

jobtype A job for which the Reports Server has an engine.

Default   `REPORT`

A.4.45 JVMOPTIONS

Table A–55 indicates which commands can use the `JVMOPTIONS` keyword.

Table A–55   Commands that can use JVMOPTIONS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	yes	yes	yes	no	no	yes

Description: Use `JVMOPTIONS` to set options for either Reports Runtime, Reports Builder, Reports Converter, or Reports Server's Java Virtual Machine (JVM). For example, you could use the following command line to start the Reports Server with a 512Mb heap space:

```
rwserver server= servername jvmoptions=-Xmx512M
```

You could also use the following command line to start Reports Builder with a 512Mb heap space:

```
rwbuilder jvmoptions=-Xmx512M
```

Multiple options can be passed, in which case the options must be enclosed in quotes:

```
rwserver server= servername jvmoptions="-Xmx256M -Xms=128M"
rwbuilder jvmoptions="-Xmx256M -Xms=128M"
```

Syntax `JVMOPTIONS={options in the Reports Runtime, Reports Builder, Reports Converter, or Reports Server's JVM}`

Default `-Xmx256M`

> **Note:**   The value set using the jvmoptions keyword will override the value in the `REPORTS_JVM_OPTIONS` environment variable, if set.

A.4.46 KILLJOBID

Table A–56 indicates which commands can use the `KILLJOBID` keyword.

Table A–56   Commands that can use KILLJOBID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use `KILLJOBID` to kill a Reports Server job with the specified job ID *n*.

Syntax `http://yourwebserver/reports/rwservlet/killjobidn[?][server=server_name][&authid=username/password][&statusformat={html|xml|xmldtd}]`

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- The job must be current (enqueued or scheduled).

- Use SHOWJOBS to see the current list of jobs. The STATUSFORMAT can be set to html (default), xml , or xmldtd to return status in that format. The status information is generated in html, xml, or xmldtd (with an internal dtd subset).

- Related keywords are SHOWJOBS , SERVER, AUTHID, and STATUSFORMAT.

- The STATUSFORMAT parameter is only valid for rwservlet, not for rwcgi.

A.4.47 LONGCHUNK

Table A–57 indicates which commands can use the LONGCHUNK keyword.

Table A–57   Commands that can use LONGCHUNK

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   LONGCHUNK is the size (in kilobytes) of the increments in which Reports 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 Oracle databases.

Syntax   LONGCHUNK=*n*

Values

n A number from 1 through 9999 (note that thousands are not expressed with any internal punctuation, e.g., a comma or a decimal point). For some operating systems, the upper limit might be lower.

Default   10

Usage Notes   LONGCHUNK can be used when running JSP-based Web reports from the command line.

A.4.48 MIMETYPE

Table A–58 indicates which commands can use the MIMETYPE keyword.

Table A–58   Commands that can use MIMETYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use MIMETYPE to override the MIME type assigned by the Reports Server when it returns output for the Web. In most cases, the default MIME type is correct, but, in cases where it is not, you can override it with this keyword. For example:

```
mimetype=application/vnd.ms-excel
```

Syntax   MIMETYPE=*string*

Values

string A valid MIME type specification.

Default   None

Usage Notes   OracleAS Reports Services does not verify the string you enter for MIMETYPE. You must ensure yourself that the string is correct for the returned report output.

A.4.49 MODE

Table A–59 indicates which commands can use the MODE keyword.

Table A–59   Commands that can use MODE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use MODE to specify whether to run the report in character mode or bitmap. This enables you to run a character-mode report from Reports Builder.

Syntax   MODE={BITMAP|CHARACTER|DEFAULT}

Values   The following values apply:

- `BITMAP` Run the report in bitmap mode.

- `DEFAULT` Run the report in the mode of the current executable being used.

- `CHARACTER` On Windows - the Reports Builder ASCII driver will be used to produce editable ASCII output.

Default `DEFAULT`

A.4.50 MODULE|REPORT

Table A–60 indicates which commands can use the MODULE|REPORT keyword.

Table A–60   Commands that can use MODULE|REPORT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use `MODULE` or `REPORT` to specify the name of the report to run.

Syntax   `REPORT|MODULE=runfile`

Values

`runfile` Any valid runfile (that is, a file with an extension of `.rep`, `.rdf`, `.jsp`, or `.xml`).

Default   None

Usage Notes

- If you specify a character-mode report, Reports Builder displays a warning, then opens the report using a page size of 8.5" x 11" and a form size of 7" x 6".

- To run the report (e.g., display it in the Paper Design View), it must be a complete report definition (i.e., contain its own data model and layout definition). You cannot run a partial report definition.

- An XML report definition *must* have an `.xml` file extension when specified with the MODULE|REPORT keyword.

- If you do not enter a file extension, the executable searches first for a file with extension `.rep`, then extension `.rdf`, then `.jsp`, and then no extension, using the file path search order to find the file.

A.4.51 NONBLOCKSQL

Table A–61 indicates which commands can use the NONBLOCKSQL keyword.

Table A–61   Commands that can use NONBLOCKSQL

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use NONBLOCKSQL to specify whether to allow other programs to execute while data is fetched from the database.

Syntax   NONBLOCKSQL={YES|NO}

Values

- YES Other programs can execute while data is being fetched.

- NO Other programs cannot execute while data is being fetched.

Default   YES

Usage Notes   NONBLOCKSQL can be used when running JSP-based Web reports from the command line.

A.4.52 NOTIFYFAILURE

Table A–62 indicates which commands can use the NOTIFYFAILURE keyword.

Table A–62   Commands that can use NOTIFYFAILURE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use NOTIFYFAILURE to specify the recipient(s) of a notification e-mail should a report request fail. Use this keyword when you configure your Reports Server to use the notification class. For more information, see the notification discussion in Configuring OracleAS Reports Services.

Syntax   NOTIFYFAILURE=*emailid*|*(emailid,emailid,...)*

Values

emailid A valid e-mail address in the form *someone@foo*.com.

Default None

Usage Notes

- The default notification e-mail templates that are used for the body of the notification e-mail are included with your installation of Oracle Application Server. The NOTIFYFAILURE template is named failnote.txt, and is located at *ORACLE_HOME*\reports\template.

- NOTIFYFAILURE can be used when running JSP-based Web reports from the command line.

A.4.53 NOTIFYSUCCESS

Table A–63 indicates which commands can use the NOTIFYSUCCESS keyword.

Table A–63 Commands that can use NOTIFYSUCCESS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description Use NOTIFYSUCCESS to specify the recipient(s) of a notification e-mail should a report request succeed. Use this keyword when you configure your Reports Server to use the notification class. For more information, see the notification discussion in **Configuring OracleAS Reports Services**.

Syntax NOTIFYSUCCESS=*emailid|(emailid,emailid,...)*

Values

emailid A valid e-mail address in the form *someone@foo*.com.

Default None

Usage Notes

- The default notification e-mail templates that are used for the body of the notification e-mail are included with your installation of Oracle Application Server. The NOTIFYSUCCESS template is named succnote.txt, and is located at *ORACLE_HOME*\reports\template.

- `NOTIFYSUCCESS` can be used when running JSP-based Web reports from the command line.

A.4.54 NUMBERFORMATMASK

Table A–64 indicates which commands can use the `NUMBERFORMATMASK` keyword.

Table A–64   Commands that can use NUMBERFORMATMASK

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use `NUMBERFORMATMASK` to specify how number values display in your delimited report output.

Syntax   `NUMBERFORMATMASK=mask`

Values   Any valid number format mask.

Default   None

Usage Notes

- This keyword can only be used if you have specified `DESFORMAT=DELIMITED` or `DESFORMAT=DELIMITEDDATA`.

    > **Note:**   For valid `NUMBERFORMATMASK` values see the *Reports Builder online help* topic, "Number Format Mask Syntax."

- `NUMBERFORMATMASK` can be used when running JSP-based Web reports from the command line.

A.4.55 OLAP_CON

Table A–65 indicates which commands can user the `OLAP_CON` keyword.

Table A–65   Commands that can use OLAP_CON

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   OLAP_CON specifies the Oracle OLAP (on-line analytical processing) connection string to connect to a database that contains multidimensional Oracle OLAP data.

Syntax   OLAP_CON=*userid/password/hostname/SID/portnumber*

Values   A valid OLAP connection string where:

- *userid* The user ID for connecting to Oracle OLAP.

- *password* The password for the user ID.

- *hostname* The host name for the database.

- *SID* The system identifier (SID) for connecting to the database.

- *portnumber* The port number for connecting to the database.

For example:

```
OLAP_Con="user1/secret1/mypc.us.oracle.com/mySID/9201
```

A.4.56 ONFAILURE

Table A–66 indicates which commands can use the ONFAILURE keyword.

Table A–66   Commands that can use ONFAILURE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use ONFAILURE to specify whether you want a COMMIT or ROLLBACK performed if an error occurs and the report fails to complete.

Syntax   ONFAILURE={COMMIT|ROLLBACK|NOACTION}

Values

- COMMIT Perform a COMMIT if the report fails.

- ROLLBACK Perform a ROLLBACK if the report fails.

- NOACTION Do nothing if the report fails.

Default

- ROLLBACK, if a USERID is provided.

- NOACTION, if called from an external source (for example, OracleAS Forms Services) with no USERID provided.

Usage Notes

- The COMMIT or ROLLBACK for ONFAILURE is performed after the report fails. Other COMMITs and ROLLBACKs can occur prior to this one. For more information, see the READONLY command.

- ONFAILURE can be used when running JSP-based Web reports from the command line.

A.4.57 ONSUCCESS

Table A–67 indicates which commands can use the ONSUCCESS keyword.

Table A–67   Commands that can use ONSUCCESS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use ONSUCCESS to specify that either a COMMIT or ROLLBACK should be performed when a report is finished running.

Syntax   ONSUCCESS={COMMIT|ROLLBACK|NOACTION}

Values

- COMMIT Perform a COMMIT when a report is done.

- ROLLBACK Perform a ROLLBACK when a report is done.

- NOACTION Do nothing when a report is done.

Default

- COMMIT, if a USERID is provided.

- NOACTION, if called from an external source (for example, OracleAS Forms Services) with no USERID provided.

Usage Notes

- 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.

- ONSUCCESS can be used when running JSP-based Web reports from the command line.

A.4.58 ORIENTATION

Table A–68 indicates which commands can use the ORIENTATION keyword.

Table A–68   Commands that can use ORIENTATION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   ORIENTATION controls the direction in which the pages of the report will print.

Syntax   ORIENTATION={DEFAULT|LANDSCAPE|PORTRAIT}

Values

DEFAULT Use the current printer setting for orientation.

LANDSCAPE Landscape orientation (long side at top and bottom).

PORTRAIT   Portrait orientation (short side at top and bottom).

Default   DEFAULT

Usage Notes

- If ORIENTATION=LANDSCAPE for a character mode report, then you must ensure that your printer definition file contains a landscape clause.

- Not supported when output to a PCL printer on Motif.

A.4.59 OUTPUTFOLDER

Table A–69 indicates which commands can use the OUTPUTFOLDER keyword.

Table A–69   Commands that can use OUTPUTFOLDER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use OUTPUTFOLDER to specify the name of the Oracle WebDB Release 2.2 or OracleAS Portal folder to push Oracle Reports output. This keyword is maintained for backward compatibility with Oracle WebDB Release 2.2 and Oracle9iAS Portal Release 1. Beginning with OracleAS Portal 10*g* (9.0.4), use OUTPUTPAGE.

Syntax   OUTPUTFOLDER=*folder*

Values

folder Any valid folder name (*internal name*) used in Oracle WebDB Release 2.2 or Oracle9iAS Portal Release 1.

Default   Oracle_Reports_Output

Usage Notes

- The value for this keyword is case sensitive.

- Use of this keyword is *optional* to push Oracle Reports output to Oracle WebDB Release 2.2 or Oracle9iAS Portal Release 1.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.60 OUTPUTIMAGEFORMAT

Table A–70 indicates which commands can use the OUTPUTIMAGEFORMAT keyword.

Table A–70   Commands that can use OUTPUTIMAGEFORMAT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use `OUTPUTIMAGEFORMAT` to choose the format for images in your report output.

Syntax   `OUTPUTIMAGEFORMAT={PNG|JPEG|JPG|GIF|BMP}`

Values

- `PNG`,`JPEG`,`JPG` Supported image formats when `DESFORMAT` value is PDF, HTML, HTMLCSS, or RTF

- `GIF` Supported image format when `DESFORMAT` value is PDF, HTML, or HTMLCSS

- `BMP` Supported image formats when `DESFORMAT` value is RTF

Usage Notes

- This command line keyword overrides the setting of the `REPORTS_ OUTPUTIMAGEFORMAT` environment variable.

- This command line keyword is not supported if the `REPORTS_DEFAULT_ DISPLAY` environment variable is explicitly set to `NO` (default is `YES`). In this case, image rendering defaults to GIF for HTML, HTMLCSS, and PDF output, and BMP for RTF output.

- You must ensure the format that you specify matches the output type. For example, BMP only works for RTF output. It will not work for HTML, HTMLCSS or PDF output.

Examples

Example 1
The following command lines generate PNG images with HTML output:

```
rwclient server=my_rep_server report=images.rdf destype=file desformat=html
desname=images.html userid=user_id outputimageformat=PNG

rwrun report=images.rdf destype=file desformat=html desname=images.html
userid=user_id outputimageformat=PNG
```

Similarly when `DESFORMAT=pdf`, images are embedded in PNG format in the generated PDF document.

Example 2

An error is displayed if an invalid value is specified for the OUTPUTIMAGEFORMAT. The following command lines generate an error message:

```
rwclient server=my_rep_server report=images.rdf destype=file desformat=html
desname=images.html userid=user_id outputimageformat=ABCD
```

```
rwrun report=images.rdf destype=file desformat=html desname=images.html
userid=user_id outputimageformat=ABCD
```

The invalid image format ABCD generates the following error message:

```
REP-35000: The image output format specified is not supported.
```

A.4.61 OUTPUTPAGE

Table A–71 indicates which commands can use the OUTPUTPAGE keyword.

Table A–71   Commands that can use OUTPUTPAGE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use OUTPUTPAGE to specify the name of the Oracle Reports page to push a report output information to. For backward compatibility with earlier versions (Oracle WebDB Release 2.2 and Oracle9iAS Portal Release 1), see OUTPUTFOLDER.

Syntax   OUTPUTPAGE=*page*

Values

page   Any valid page name (*internal name*) used in Oracle9iAS Portal Release 1.

Default   Oracle_Reports_Output

Usage Notes

- Use of this keyword is *optional* to push Oracle Reports output to Oracle9iAS Portal Release 1:

  - If you do not specify an output page, OracleAS Portal will create a default page named Oracle_Reports_Output.

- If you specify an output page, use the internal name and not the display name. The internal name is used to uniquely identify the OracleAS Portal component instance.

- The value for this keyword is *case sensitive*.

- The page should contain at least one item region when used with `DESTYPE=ORACLEPORTAL.`

- Relevant keywords include `CONTENTAREA*`, `EXPIREDAYS`, `ITEMTITLE`, `OUTPUTFOLDER*`, `OUTPUTPAGE`, `PAGEGROUP`, `REPLACEITEM`, `SITENAME*`, `STATUSFOLDER*`, `STATUSPAGE`.

  * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.62 OVERWRITE

Table A–72 indicates which commands can use the OVERWRITE keyword.

Table A–72   Commands that can use OVERWRITE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use OVERWRITE to specify whether to overwrite existing files with the converted files.

Syntax   `OVERWRITE={YES|NO|PROMPT}`

Values

- `YES` Automatically overwrite any existing files of the same name.

- `NO` Do not to convert reports if there are existing files of the same name and display a warning message

- `PROMPT` Prompt you before overwriting any existing files.

Default   `NO`

A.4.63 P_AVAILABILITY

Table A–73 indicates which commands can use the P_AVAILABILITY keyword.

Table A–73   Commands that can use P_AVAILABILITY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_AVAILABILITY to specify the name of the availability calendar that determines when the reports specified will be available for processing. This keyword is only used when DTYPE=REGISTER.

Syntax   P_AVAILABILITY=*calendar_name*

Values

calendar_name Any valid availability calendar name.

Default   None

Usage Notes   The availability calendar must exist in OracleAS Portal before running the SQL*PLUS script. If it does not, an invalid package may be created.

A.4.64 P_DESCRIPTION

Table A–74 indicates which commands can use the P_DESCRIPTION keyword.

Table A–74   Commands that can use P_DESCRIPTION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_DESCRIPTION to specify the text that provides additional information about the report. This keyword is only used when DTYPE=REGISTER.

Syntax   P_DESCRIPTON=*description_text*

Values

description_text Any text string.

Default   None

A.4.65 P_FORMATS

Table A–75 indicates which commands can use the P_FORMATS keyword.

Table A–75   Commands that can use P_FORMATS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_FORMATS to specify the allowable destination formats for the specified reports. This keyword is only used when DTYPE=REGISTER.

Syntax   P_FORMATS=*destination_format|(destination_format1,destination_format2,...)*

Values

destination_format Any valid destination format (e.g., HTML), or a list of valid destination formats enclosed by parentheses with a comma separating the names (e.g., HTMLCSS,PDF,RTF).

Default   None

A.4.66 P_NAME

Table A–76 indicates which commands can use the P_NAME keyword.

Table A–76   Commands that can use P_NAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_NAME to specify the report name displayed in OracleAS Portal. This keyword is only used when DTYPE=REGISTER.

Syntax   P_NAME=*report_name*

Values

report_name Any report name.

Default   If P_NAME is not specified, the PL/SQL function is populated with the report definition file name.

Usage Notes

- If P_NAME is not specified, the PL/SQL function is populated with the report definition file name.

- Specify P_NAME only when you want to use the same report name for each report definition file being registered in OracleAS Portal. This option is typically left blank.

- The report name cannot be prefaced with numeric characters (e.g., 401K_ report is an invalid file name and my_401K_report is valid).

A.4.67 P_OWNER

Table A–77 indicates which commands can use the P_OWNER keyword.

Table A–77   Commands that can use P_OWNER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_OWNER to specify the OracleAS Portal DB Provider that owns a report's package, which is created when the report definition files are registered. This keyword is only used when DTYPE=REGISTER.

Syntax   P_OWNER=*portal_dbprovider*

Values

portal_dbprovider Any valid OracleAS Portal DB Provider name.

Default   The name of the OracleAS Portal DB Provider to which you are connected when you run the SQL*PLUS script file.

A.4.68 P_PFORMTEMPLATE

Table A–78 indicates the commands that can use the P_PFORMTEMPLATE keyword.

Table A–78   Commands that can use P_PFORMTEMPLATE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_PFORMTEMPLATE to specify the name of the OracleAS Portal template that determines the style of the Runtime Parameter Form. This keyword is only used when DTYPE=REGISTER.

Syntax   P_PFORMTEMPLATE=*template_name*

Values

template_name Any valid OracleAS Portal template name.

Default   None

A.4.69 P_PRINTERS

Table A–79 indicates the commands that can use the P_PRINTERS keyword.

Table A–79   Commands that can use P_PRINTERS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_PRINTERS to specify the allowable printers for the specified reports. This keyword is only used when DTYPE=REGISTER.

Syntax   P_PRINTERS=*printer_name*

Values

printer_name Any valid printer (e.g., PRT1), or a list of valid printers enclosed by parentheses with a comma separating the names (e.g., (PRT1,PRT2,PRT3)).

Default   None

Usage Note   Access to the printer(s) should already exist in OracleAS Portal before running the SQL*Plus script.

A.4.70 P_PRIVILEGE

Table A–80 indicates which commands can use the P_PRIVILEGE keyword.

Table A–80   Commands that can use P_PRIVILEGE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_PRIVILEGE to specify the users or roles who have access privileges to run the specified reports. This keyword is only used when DTYPE=REGISTER.

Syntax   P_PRIVILEGE=*user_name*

Values

user_name Any user name or role that OracleAS Portal can recognize (e.g., SCOTT), or a list of user names or roles enclosed by parentheses with a comma separating the names (e.g., (SCOTT,JABERS,PMARTIN)).

Default   None

A.4.71 P_SERVERS

Table A–81 indicates which commands can use the P_SERVERS keyword.

Table A–81   Commands that can use P_SERVERS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_SERVERS to specify the names of the restricted Reports Servers that can run the report. This keyword is only used when DTYPE=REGISTER.

Syntax   P_SERVERS=*tnsname*

Values

tnsname Any valid TNS name of the Reports Server (e.g., repserver), or a list of valid Reports Server TNS names enclosed by parentheses with a comma separating the names (e.g., repserver, acct_server, sales_server).

Default   None

Usage Notes   Access to the Reports Server(s) should already exist in OracleAS Portal.

A.4.72 P_TRIGGER

Table A–82 indicates the commands that can use the P_TRIGGERS keyword.

Table A–82   Commands that can use P_TRIGGER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_TRIGGER to specify the PL/SQL function that is executed when parameter values are specified on the command line and when users accept the Runtime Parameter Form. The function must return a boolean value (TRUE or FALSE). For example:

```
P_TRIGGER=Is begin IF UPPER(DESTYPE) = 'PRINTER' AND EMPNAME = 'SMITH' THEN
RETURN(TRUE); ELSE RETURN(FALSE); END IF; end;
```

This keyword is only used when DTYPE=REGISTER.

Syntax   P_TRIGGER=*plsql_function*

Values

plsql_function   Any valid PL/SQL function that returns a boolean value.

Default   None

A.4.73 P_TYPES

Table A–83 indicates which commands can use the P_TYPES keyword.

Table A–83   Commands that can use P_TYPES

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use P_TYPES to specify the allowable destination types for the specified reports. This keyword is only used when DTYPE=REGISTER.

Syntax   P_TYPES=*destination_type*

Values

destination_type Any valid destination type (e.g., CACHE), or a list of valid destination types enclosed by parentheses with a comma separating the names (e.g., CACHE,MAIL,PRINTER).

Default   None

A.4.74 PAGEGROUP

Table A–84 indicates which commands can use the PAGEGROUP keyword.

Table A–84   Commands that can use PAGEGROUP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use PAGEGROUP to specify the name of the Oracle9iAS Portal Release 1 page group to which report output should be pushed. For backward compatibility with earlier versions, see SITENAME (for Oracle WebDB Release 2.2) and CONTENTAREA (for Oracle9iAS Portal Release 1). The page group must be created in Oracle9iAS Portal Release 1 before you can use this keyword.

Syntax   PAGEGROUP=*pagegroup*

Values

pagegroup Any valid page group name (*internal name*) used in Oracle9iAS Portal Release 1.

Default   None

Usage Notes

- Use of this keyword is *required* to push Oracle Reports output to Oracle9iAS Portal Release 1.

- The page group name should be the internal name and *not* the display name. The internal name is used to uniquely identify the Oracle9iAS Portal Release 1 page instance.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

    * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.75 PAGESIZE

Table A–85 indicates which commands can use the PAGESIZE keyword.

Table A–85   Commands that can use PAGESIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	yes	yes	yes	no

Description   Use PAGESIZE to set 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 more than zero. The maximum width and height depends which unit of measurement was set in Reports Builder (**Edit > Preferences > General** tab). For inches, the maximum width and height is 512 inches. For centimeters, it is 1312 centimeters. For picas, it is 36,864 picas.

Default   For bitmap, 8.5 x 11 inches. For character mode, 80 x 66 characters. If the report was designed for character mode and is being run or converted on bitmap, then the following formula is used to determine page size if none is specified: (default page size * character page size)/default character page size. For example, if the character page size is 80 x 20, then the bit-mapped page size would be: ((8.5 * 80)/80) x ((11 * 20)/66) = (680/80) x (220/66) = 8.5 x 3.33.

Usage Notes

- On some printers the printable area of the physical page is restricted. For example, the sheet of paper a printer takes 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 Reports 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.

- Letter size is 8.5 inches x 11 inches. A4 size is 210mm x 297mm, or 8.25 inches x 11.75 inches.

- If you use the PAGESIZE keyword, 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.

A.4.76 PAGESTREAM

Table A–86 indicates which commands can use the PAGESTREAM keyword.

Table A–86   Commands that can use PAGESTREAM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   PAGESTREAM enables or disables page streaming (pagination) 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 Paginate the report output.

- NO Output the report without pagination.

Default   NO

A.4.77 PARAMFORM

Table A–87 indicates which commands can use the PARAMFORM keyword.

Table A–87   Commands that can use PARAMFORM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use PARAMFORM to specify whether to display the Runtime Parameter Form when you execute a report via CGI or a servlet. PARAMFORM is used only to supply parameters to paper layout reports, not Web source reports.

Syntax   PARAMFORM=YES|NO|HTML

Values

- YES Display the Parameter Form.

- NO Do not display the Parameter Form.

- HTML Display the Parameter Form in HTML format.

Default   NO

Usage Notes

- PARAMFORM=YES is incompatible with BATCH=YES because it is not meaningful to have the Runtime Parameter Form appear in batch mode.

- Do not use this keyword when running a report in an OracleAS Portal environment. This is because OracleAS Portal allows you to set up a Reports runtime parameter form, which may conflict with a form you specify with the PARAMFORM keyword.

A.4.78 PARSEQUERY

Table A–88 indicates which commands can use the PARSEQUERY keyword.

Table A–88   Commands that can use PARSEQUERY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use PARSEQUERY to parse an rwservlet query and display the constructed Reports Server command line.

Syntax
```
http://yourwebserver/reports/rwservlet/parsequery[?][server=servername][&authid=
username/password]
```

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- Related keywords are SERVER and AUTHID.

A.4.79 PDFCOMP

Table A–89 indicates which commands can use the PDFCOMP keyword.

Table A–89   Commands that can use PDFCOMP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use PDFCOMP to specify whether PDF output should be compressed.

Syntax   PDFCOMP=value|{YES|NO}

Values

- value Any value 0 though 9. A value of 0 means PDF output will not be compressed. A value of 1 through 9 will compress the PDF output and permit users to control the compression level.

- YES Compresses output at compression level 6.

- NO Compresses output at compression level 0 (no compression).

Default   6

A.4.80 PDFEMBED

Table A–90 indicates which commands can use the PDFEMBED keyword.

Table A–90   Commands that can use PDFEMBED

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description　Use PDFEMBED to specify whether Oracle Reports will embed the Type1 PostScript font file(s) specified in the uifont.ali file into PDF output.

Syntax　PDFEMBED={YES|NO}

Values

- YES The PDF driver will embed the font(s) specified in the [PDF:Embed] header of the uifont.ali file into the PDF output.

- NO The font(s) will not be added to PDF output.

Default　YES

A.4.81 PRINTJOB

Table A–91 indicates which commands can use the PRINTJOB keyword.

Table A–91   Commands that can use PRINTJOB

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	yes	no	no	no	no

Description　Use PRINTJOB to specify whether the **Print Job** dialog box should be displayed before running a report.

Syntax　PRINTJOB={YES|NO}

Values

- YES The **Print Job** dialog box is displayed before the report is run.

- NO The report is run without displaying the **Print Job** dialog box.

Default   NO

Usage Notes

- When a report is run as a spawned process (that is, one executable, such as rwrun, is called from within another executable, such as rwbuilder), 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.

A.4.82 READONLY

Table A–92 indicates which commands can use the READONLY keyword.

Table A–92   Commands that can use READONLY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use READONLY to request read consistency across multiple queries in a report. When accessing data from Oracle 9*i*, read consistency is accomplished by a SET TRANSACTION READ ONLY statement.

> **Note:**   Refer to *Oracle9i* SQL documentation (available on the Oracle Technology Network, (http://otn.oracle.com) for more information on SET TRANSACTION READ ONLY.

Syntax   READONLY={YES|NO}

Values

YES Requests read consistency.

NO Do not provide read consistency.

Default   NO

Usage Notes

- This keyword is only useful for reports using multiple queries. Oracle 9*i* automatically provides read consistency, without locking, for single query reports.

- In the Report trigger order of execution, SET TRANSACTION READ ONLY must be set up before the data fetch occurs.

- READONLY can be used when running JSP-based Web reports from the command line.

A.4.83 RECURSIVE_LOAD

Table A–93 indicates which commands can use the RECURSIVE_LOAD keyword.

Table A–93   Commands that can use RECURSIVE_LOAD

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	yes	yes	yes	no

Description   Use RECURSIVE_LOAD to specify whether to validate external references in program units when running a report. If any of the references become invalid, the program unit is automatically recompiled. Setting RECURSIVE_LOAD **to** NO is useful when running your report against a different database than the one against which its PL/SQL was originally compiled.

Syntax   RECURSIVE_LOAD={YES|NO}

Values

- YES Validates external references when running a report. If any of the references become invalid, the program unit is recompiled (whether it be in .rdf or .pll).

- NO Does not validate external references when running a report. This setting is useful when running a report against a different database than the one against which its PL/SQL was originally compiled.

Default   YES

A.4.84 REPLACEITEM

Table A–94 indicates which commands can use the REPLACEITEM keyword.

Table A–94   Commands that can use REPLACEITEM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use REPLACEITEM to specify that the current report output being pushed to Oracle9iAS Portal Release 1 should replace an earlier version of the same item stored in the same output target.

Syntax   REPLACEITEM={YES|NO}

Values

- YES Earlier report output should be replaced.

- NO Do not replace previous version. In this case, a link to the new output will be added to a list of links to previous versions of the same report.

Default   None

Usage Notes

- Use of this keyword is *optional* to push Oracle Reports output to Oracle9iAS Portal Release 1.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

   * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.85 REPLYTO

Table A–95 indicates which commands can use the REPLYTO keyword.

Table A–95   Commands that can use REPLYTO

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use REPLYTO to specify the e-mail address to which replies should be sent when the sender wants replies to go to someone other than the sender (specified by the FROM keyword).

Syntax   REPLYTO=*emailid*

Values

emailid A valid e-mail address in the form of *someone@foo*.com.

Default   None

Usage Notes   Related keywords include BCC, CC, FROM, REPLYTO, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.4.86 REPORT|MODULE

See MODULE | REPORT.

A.4.87 ROLE

Table A–96 indicates which commands can use the ROLE keyword.

Table A–96   Commands that can use ROLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use ROLE to specify the database role to be checked for the report at runtime. This keyword is useful for giving you the ability to run reports that query database tables to which you would not normally have access privileges.

Syntax   ROLE=*{rolename[/rolepassword]}*

Values

- *rolename* A valid role.

- *rolepassword* (Optional) The matching role password.

Default   None

Usage Notes ROLE can be used when running JSP-based Web reports from the command line.

A.4.88 RUNDEBUG

Table A–97 indicates which commands can use the RUNDEBUG keyword.

Table A–97   Commands that can use RUNDEBUG

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description Use RUNDEBUG to specify that you want extra runtime checking for logical errors in the report. RUNDEBUG checks for things that are not errors but might result in undesirable output, and displays these as warnings at runtime, before displaying the report output. Using RUNDEBUG to run a report in debug mode is not the same as debugging a report using the PL/SQL Interpreter.

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. Such objects will be fixed in size regardless of the Vertical Elasticity and Horizontal Elasticity property settings.

- Bind variables referenced at the wrong frequency in PL/SQL.

Syntax RUNDEBUG={YES|NO}

Values

- YES Perform extra runtime error checking.

- NO Do not perform extra runtime error checking.

Default YES

Usage Notes RUNDEBUG can be used when running JSP-based Web reports from the command line.

A.4.89 SAVE_RDF

Table A–98 indicates which commands can use the SAVE_RDF keyword.

Table A–98   Commands that can use SAVE_RDF

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	yes	yes	no	no	no	no

Description   Use SAVE_RDF to specify a filename for a combined RDF file and XML customization file. This keyword is useful when you combine an existing RDF file with a Oracle Reports XML customization file using the CUSTOMIZE keyword, and you wish to save the combination to a new RDF file.

Syntax   SAVE_RDF=*filename*.rdf

Values

filename.rdf   Any valid file name.

Default   None

Usage Notes   You can use SAVE_RDF with a JSP file, but only the paper layout part, not the Web source.

A.4.90 SCHEDULE

Table A–99 indicates which commands can use the SCHEDULE keyword.

Table A–99   Commands that can use SCHEDULE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description   Use SCHEDULE to set the day, time, and frequency a report should be run. The default is to run the report once, now. Time values are expressed according to a 24-hour day (i.e., one o'clock is expressed 13:00). To eliminate the need for quoting the scheduling command, use underscores (_) instead of spaces. You can also specify an expiration for a report job after a number of runs or on a particular date/time. For example, use:

```
SCHEDULE=every_first_fri_of_month_from_15:53_Oct_23,_1999_retry_3_after_1_hour_
expires_on_15:53_Oct_23,_2003
```

```
SCHEDULE=last_weekday_before_15_from_15:53_Oct_23,_1999_retry_after_1_hour_
expires_after_100
```

Or:

```
SCHEDULE="every first fri of month from 15:53 Oct 23, 1999 retry 3 after 1 hour
expires on 15:53 Oct 23, 2003"
SCHEDULE="last weekday before 15 from 15:53 Oct 23, 1999 retry after 1 hour
expires after 100"
```

Syntax   SCHEDULE=*string*

where the *string* is:

[*FREQ* from] *TIME* [retry {*n*} after *LEN* expires {on|after} time|*n*]

Table A–100 lists and explains the values used in this string.

Table A–100   Values for string used with the SCHEDULE keyword

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
EXPIRES	on {today	tomorrow	{*MONTHS* {*d*	*dd*} [,year]}}	after *n*

Default   None

A.4.91 SERVER

Table A–101 indicates which commands can use the SERVER keyword.

Table A–101   Commands that can use SERVER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	yes

Description   Use SERVER to specify the name of the Reports Server you want to use to run this report.

Syntax   SERVER=*servername*

Values   The server name or a TNS service entry name if you're using a 6*i* compatible server.

Default   The server name specified in the REPORTS_SERVER environment variable for rwcgi.

Usage Notes

- For jobs run with rwcgi, you can set the REPORTS_SERVER environment variable on your Web server machine and omit the SERVER keyword to process requests using the default server, or you can include the SERVER keyword to override the default. For jobs run with rwservlet or as a JSP, you can omit the SERVER keyword if you have specified a default server in the servlet configuration file, rwservlet.properties; or you can include the SERVER keyword to override the default.

- SERVER can be used when running JSP-based Web reports from the command line.

A.4.92 SHOWAUTH

Table A–102 indicates which commands can use the SHOWAUTH keyword.

Table A–102   Commands that can use SHOWAUTH

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use SHOWAUTH to display the Reports Server logon page and runs the report .

Syntax
http://*yourwebserver*/reports/rwservlet/showauth[?][server=*servername*][&authid=*us
ername*/*password*]][&nextpage=run report URL][&authtype={s|d}]

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- When `authtype=s` the **Reports System User Authentication** dialog box is displayed. The **Reports Database User Authentication** dialog box is displayed when `authtype=d`.

- Related keywords are SERVER and AUTHID.

A.4.93 SHOWENV

Table A–103 indicates which commands can use the SHOWENV keyword.

Table A–103   Commands that can use SHOWENV

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use SHOWENV to display the `rwservlet` configuration file (`rwservlet.properties`).

Syntax
```
http://yourwebserver/reports/rwservlet/showenv[?][server=servername][&authid=use
rname/password]
```

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- Related keywords are SERVER and AUTHID.

A.4.94 SHOWJOBID

Table A–104 indicates which commands can use the SHOWJOBID keyword.

Table A–104 Commands that can use SHOWJOBID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	no	no

Description   SHOWJOBID shows the status of the Reports Server job with job ID *n*.

Syntax   http://*yourwebserver*/reports/rwservlet/showjobid*n*[?][server=*server_name*][&authid=*username/password*][&statusformat={html|xml|xmldtd}]

Values   See Syntax.

Default   None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- The job must be current (enqueued or scheduled).

- Use SHOWJOBS to see the current list of jobs. The STATUSFORMAT can be set to html (default), xml , or xmldtd to return status in that format. The status information is generated in html, xml, or xmldtd (with an internal dtd subset).

- Related keywords are SHOWJOBS , SERVER, AUTHID, and STATUSFORMAT.

- The STATUSFORMAT parameter is only valid for rwservlet, not for rwcgi.

A.4.95 SHOWJOBS

Table A–105 indicates which commands can use the SHOWJOBS keyword.

Table A–105   Commands that can use SHOWJOBS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use SHOWJOBS to display a Web view of Reports Server queue status.

Syntax   http://*yourwebserver*/*reports*/rwservlet/showjobs[*n*][?][server=*server_name*][&authid=*username/password*][&queuetype=current | past | future][&startrow=start_position_in_job_queue][&count=number_of_jobs_to_

```
display][&statusformat={html|xml|xmldtd}]
```

Values: See Syntax

Default None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- The name of the Reports Server must be specified implicitly by environment variable or servlet configuration file, or explicitly in the URL request. The refresh number *n* is optional. When it is specified, the report's queue status will be updated every *n* seconds.

- The STATUSFORMAT can be set to html (default), xml , or xmldtd to return status in that format. The status information is generated in html, xml, or xmldtd (with an internal dtd subset).

- The QUEUETYPE, STARTROW, COUNT, and STATUSFORMAT parameters are valid only for rwservlet and not for rwcgi.

- Related keywords are SERVER, AUTHID, and STATUSFORMAT.

A.4.96 SHOWMAP

Table A–106 indicates which commands can use the SHOWMAP keyword.

Table A–106   Commands that can use SHOWMAP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	yes	no

Description   Use SHOWMAP to display rwservlet key mappings.

Syntax
```
http://yourwebserver/reports/rwservlet/showmap[?][server=servername][&authid=use
rname/password]
```

Values See Syntax.

Default None

Usage Notes:

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- Related keywords are SERVER and AUTHID.

A.4.97 SHOWMYJOBS

Table A–107 indicates which commands can use the SHOWMYJOBS keyword.

Table A–107 Commands that can use SHOWMYJOBS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	no	no

Description Use SHOWMYJOBS to display the Reports Server queue status for a particular user.

Syntax http://*yourwebserver*/reports/rwservlet/showmyjobs[?][server=s*erver_name*][&authid=*username*/*password*][&statusformat={html|xml|xmldtd}]

Values See Syntax.

Default None

Usage Notes

- This keyword is a command that does not require a value; i.e., commands are entered by themselves without a corresponding value.

- The STATUSFORMAT can be set to html (default), xml , or xmldtd to return status in that format. The status information is generated in html, xml, or xmldtd (with an internal dtd subset).

- Related keywords are SERVER, AUTHID, and STATUSFORMAT.

- The STATUSFORMAT parameter is only valid for rwservlet, not for rwcgi.

A.4.98 SHUTDOWN

Table A–108 indicates which commands can use the SHUTDOWN keyword.

Table A–108   Commands that can use SHUTDOWN

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	no	no	yes

Description   Use SHUTDOWN to shut down a previously running server. You must also use AUTHID to supply a user name and password.

Syntax   SHUTDOWN={NORMAL|IMMEDIATE}

Values   NORMAL or IMMEDIATE

Default   NORMAL

Usage Notes   The user of the SHUTDOWN keyword must be an Oracle Reports Administrative user. If the server has security enabled, it will query the security API to determine the user's role eligibility to execute the shutdown (in other words, the user must be an Oracle Reports Administrative user). If security is not enabled, then the user must nonetheless be an Oracle Reports Administrative user defined for that server.

A.4.99 SITENAME

Table A–109 indicates which commands can use the SITENAME keyword.

Table A–109   Commands that can use SITENAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use SITENAME to specify the name of the WebDB site to push a report's output to. This keyword is maintained for backward compatibility with Oracle WebDB Release 2.2; for backward compatibility with Oracle9iAS Portal Release 1, see CONTENTAREA. Beginning with OracleAS Portal Release 2, use PAGEGROUP.

Syntax   SITENAME=*name*

Values

name Any valid site name used in Oracle WebDB Release 2.2.

Default   None

Usage Notes

- Use of this keyword is *required* to push Oracle Reports output to Oracle9iAS Portal Release 1.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.100 SOURCE

Table A–110 indicates which commands can use the SOURCE keyword.

Table A–110   Commands that can use SOURCE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use SOURCE to specify the report/library or list of reports/libraries to be converted. The rwconverter command requires that you specify a source report or library.

Syntax   SOURCE={*source_name*|(*source_name1*, *source_name2*, …)}

Values   Any valid report/library name or filename, or a list of valid report/library names or filenames enclosed in parentheses and separated by commas (e.g., (qanda, test, dmast)).

Default   None

Usage Notes

- SQL wildcard characters (% and _) may be used for reports or libraries that are stored in the database. For example, R% would fetch all reports stored in the database that begin with R. All reports that match will be converted.

- A list of report/library names or filenames must be enclosed in parentheses, with commas separating the names. For example:

```
(qanda,test,dmast) OR (qanda, test, dmast)
```

- Wildcard characters are invalid for reports/libraries stored in files (i.e., with extensions of rdf, rep, rex, pld, pll, or xml).

- The value(s) for the SOURCE keyword may be operating system-specific.

- If you are using user-owned Reports Builder tables, reports/libraries from multiple users must be converted for each user individually.

- To convert reports/libraries, you must have created them or been granted access to the ones you did not create. If no userid is prefixed to the report/library name, the userid is assumed to be the current user.

- When DTYPE=REGISTER, you may only want to list report definition files with common parameters, such as destination types and formats, user access, and availability calendars.

A.4.101 SQLTRACE

Table A–111 indicates which commands can use the SQLTRACE keyword.

Table A–111   Commands that can use SQL_TRACE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use SQLTRACE to specify whether to perform SQL tracing on your report without modifying the report definition.

Syntax   SQLTRACE=[YES|NO]

Values

- YES SQL tracing will be performed on the report.

- NO SQL tracing will not be performed on the report.

Default   NO

A.4.102 SSOCONN

Table A–112 indicates which commands can use the SSOCONN keyword.

Table A–112   Commands that can use SSOCONN

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	no	no

Description   Use SSOCONN to specify one or more connect strings to use to connect to one or more data sources in a Single Sign-On environment.

Syntax   SSOCONN=key[/type[/conn_str]][,key[/type[/conn_str]]]

For example:

```
ssoconn=mykey/OracleDB/userid
```

Values   The following information describes the variable values expressed in the SSOCONN syntax:

- *key* refers to a connection string value stored in the Oracle Internet Directory (OID).

- *type* is a predefined character string that specifies a Reports data source type. Types provided with your Oracle Application Server installation include OracleDB, JDBCPDS (Java Database Connectivity Pluggable Data Source), and ExpressPDS (Oracle Express Pluggable Data Source).

- *conn_str* is the name of the connection string parameter that rwservlet will compose to run the report

Default   None

Usage Notes

- If multiple data sources are used in the report, use a comma to separate data source connection strings. For example:

  ```
  ssoconn=key1/type1/conn_str,key2/type2/conn_str2,key3/type3/conn_str3
  ```

- Simplified versions of the SSOCONN option are also available. Table A–113 provides examples.

- When you use SSOCONN in a command line, you cannot:

  - Specify authid in the same command line.

  - Run against a Reports Server that is not secure.

■ Have `SINGLESIGNON` set to `NO` in `rwservlet.properties`.

Doing any of the above with `SSOCONN` in the command line results in an error.

Table A–113   Simplified versions of the SSOCONN option

option	Description
ssoconn=mkey	When only the key name is specified, the default type (Oracle DB) will be used, and the connect parameter is the USERID keyword.
ssoconn=mkey/PDSApp	When both key name and application type are specified, the connection parameter is the USERID keyword.
ssoconn=mkey/PDSApp/P_1	When everything is specified, the specified values are used.

■ `SSOCONN` can be used when running JSP-based Web reports from the command line. If you are using the `SSOCONN` keyword with a report run as a JSP, use an ampersand(&) to separate connection strings. For example:

```
http://…:8888/myjsp/foo.jsp?name1=value1&name2=value2 …
```

> **Note:** For more information about Oracle Reports and Single Sign-On (SSO), see Configuring and Administering OracleAS Single Sign-On.

A.4.103 STATUSFOLDER

Table A–114 indicates which commands can use the `STATUSFOLDER` keyword.

Table A–114   Commands that can use STATUSFOLDER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use `STATUSFOLDER` to specify the name of the Oracle WebDB Release 2.2 or Oracle9iAS Portal Release 1 folder to push status information into. If this is omitted, a new folder is created called `Oracle_Reports_Status`. This keyword is maintained for backward compatibility with Oracle WebDB Release 2.2 and Oracle9iAS Portal Release 1. Beginning with OracleAS Portal Release 2, use `STATUSPAGE`.

Syntax STATUSFOLDER=*folder*

Values

folder Any valid folder name (*internal name*) used in Oracle WebDB Release 2.2 or Oracle9iAS Portal Release 1.

Default Oracle_Reports_Status

Usage Notes

- Use of this keyword is *optional* to push Oracle Reports output to Oracle WebDB Release 2.2 or Oracle9iAS Portal Release 1.

- The value for this keyword is case sensitive.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

    * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.104 STATUSFORMAT

Table A–115 indicates which commands can use the STATUSFORMAT keyword.

Table A–115   Commands that can use STATUSFORMAT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	yes	no	no

Description Use STATUSFORMAT to specify the format for the Web view of Reports Server queue status.

Syntax http://*yourwebserver*/rwservlet/showjobs?server=*server_name*&statusformat={html|xml|xmldtd}

Values

- HTML Specify that the Reports queue status output should be in HTML format.

- XML Specify that it should be in XML format.

- XMLDTD Specify that it should be in XML format with in-line Data Type Definition information.

Default HTML

Usage Notes Use STATUSFORMAT in conjunction with the SHOWJOBS and SHOWMYJOBS keywords.

A.4.105 STATUSPAGE

Table A–116 indicates which commands can use the STATUSPAGE keyword.

Table A–116 Commands that can use STATUSPAGE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description Use STATUSPAGE to specify the name of the Oracle9iAS Portal Release 1 page to push job status information into. If this is omitted, a new page is created called Oracle_Reports_Status. For backward compatibility with earlier versions (Oracle WebDB Release 2.2 and Oracle9iAS Portal Release 1), see STATUSFOLDER.

Syntax STATUSPAGE=*page*

Values

page Any valid page name (*internal name*) used in Oracle9iAS Portal Release 1.

Default Oracle_Reports_Status

Usage Notes

- Use of this keyword is *optional* to push Oracle Reports output to Oracle9iAS Portal Release 1.

- The value for this keyword is case sensitive.

- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, REPLACEITEM, SITENAME*, STATUSFOLDER*, STATUSPAGE.

   * maintained for backward compatibility with Oracle9iAS Portal Release 1 and/or Oracle WebDB Release 2.2.

A.4.106 STYPE

Table A–117 indicates which commands can use the STYPE keyword.

Table A–117   Commands that can use STYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use STYPE to specify the format of the report(s) or libraries to be converted.

Syntax   STYPE={PLDFILE|PLLFILE|RDFFILE|REXFILE|XMLFILE|JSPFILE}

Values   Use any one of the following values:

- PLDFILE Source PL/SQL libraries are stored in files in ASCII format.

- PLLFILE Source PL/SQL libraries are stored in files containing source code and P-code (compiled PL/SQL).

- RDFFILE Source report(s) are stored in one or more report definition files (files with the rdf extension).

- REXFILE Source report(s) are stored in one or more text files (files with the rex extension).

- XMLFILE Source report(s) are stored in one or more XML files.

- JSPFILE Source report(s) are stored in one or more JSP files.

Default   RDFFILE

Usage Notes   When DTYPE=REGISTER, choose RDDFILE, REXFILE, XML, or JSPFILE for STYPE.

A.4.107 SUBJECT

Table A–118 indicates which commands can use the SUBJECT keyword.

Table A–118   Commands that can use SUBJECT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	no	no	yes	yes	no

Description   Use SUBJECT to specify the subject line of an e-mail.

Syntax   SUBJECT=*any text*

Values   Any text.

Default   None

Usage Notes

- Enclose subjects that contain character spaces in quotation marks (" "). Single-word subjects do not require quotation marks.

- Related keywords include BCC, CC, FROM, REPLYTO, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.4.108 SUPPRESSLAYOUT

Table A–119 indicates which commands can use the SUPPRESSLAYOUT keyword.

Table A–119   Commands that can use SUPPRESSLAYOUT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   Use SUPPRESSLAYOUT to specify whether to suppress the formatting of the paper layout at runtime. The keyword allows users to control whether the paper layout in a report is executed at runtime. The most common use of this keyword is to increase the performance of JSP reports. Since a JSP report may have a paper layout and reference objects in it via an <rw:include> tag, Oracle Reports formats the paper layout before running the JSP section of the report. To improve the performance of single source JSP reports that store both paper and Web layouts but do not reference paper layout objects, set SUPPRESSLAYOUT=YES on the command line.

> **Note:**   If there is an <rw:include> tag, then no output will be created for the tag.

Syntax   SUPPRESSLAYOUT=[YES|NO]

Values

- YES Paper layout objects will not be formatted at runtime

- NO Paper layout objects will be formatted at runtime.

Default   NO

A.4.109 TOLERANCE

Table A–120 indicates which commands can be used with the TOLERANCE keyword.

Table A–120   Commands that can use TOLERANCE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description   Use TOLERANCE to set the maximum acceptable time (in minutes) 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 Reusing Report Output from Cache for more information on duplicate job detection.

Syntax   TOLERANCE=*time_string*

Values

time_string Can be in one of two formats:

- *n*{*unit*}, for a number with an optional unit. The unit can be minute(s), hour(s), or day(s). The default unit is minute(s) if no unit is specified.

- {*Mon DD*, *YYYY*} *hh*:*mi*:*ss* am|pm {*timezone*}, for a date/time format. Date information is optional. If it isn't specified, *today* is assumed. Time zone is also optional. If it isn't specified, the Reports Server's timezone is used. The date/time is always in a US locale. This format is the same as defined in the Java DateFormat.MEDIUM type.

Default   None

Usage Notes

- If TOLERANCE is not specified, then OracleAS 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 OracleAS Reports Services reuses the output of the currently running job even if TOLERANCE is not specified or is set to zero.

A.4.110 TRACEFILE

Table A–121 indicates which commands can use the TRACEFILE keyword.

Table A–121   Commands that can use TRACEFILE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	no	no	no

Description   TRACEFILE is the name of the file in which trace information is logged.

> **Note:**   See Report Trace for additional information about how tracing works with Oracle Reports.

Syntax   TRACEFILE=*tracefile*

Values   Any valid file name.

Default   None

Usage Notes

- In a runtime environment, or when you are monitoring Reports Services components, you can use all three TRACE keywords. But for server security, in a server environment you can use only TRACEOPTS. This is to prevent files from being written arbitrarily to the Reports Server's file system.

- Tracing for the Reports Server is configured in the server configuration file, *server_name*.conf (see Configuring OracleAS Reports Services). Tracing for the Reports Servlet is configured in the servlet configuration file, rwservlet.properties (see Configuring OracleAS Reports Services). Tracing for individual jobs is specified from the runtime command line, via the TRACEOPTS command line option.

- Whether an existing file will be overwritten or appended to depends on the TRACEMODE setting.

- Trace information can only be generated when running a `.rdf` file. You cannot specify logging when running a .rep file.

- If you specify `TRACEOPTS=TRACELOG` or `TRACEOPTS=TRACEERR` as well as `TRACEFILE`, all of the trace information will be placed in the most recently specified file. For example, in the following case, all of the specified trace information would be placed in `err.log` because it is the last file specified in the RWRUN command.

```
RWRUN MODULE=order_entry USERID=scott/tiger
TRACEFILE=trace.log LOGFILE=mylog.log
ERRFILE=err.log
```

A.4.111 TRACEMODE

Table A–122 indicates which commands can use the TRACEMODE keyword.

Table A–122   Commands that can use TRACEMODE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	no

Description   TRACEMODE specifies whether to append new trace information to existing information in the file specified by TRACEFILE, 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 Notes

- In a runtime environment, or when you are monitoring Reports Services components, you can use all three TRACE keywords. But for server security, in a server environment you can use only TRACEOPTS. This is to prevent files from being written to the server.

- Tracing for the Reports Server is configured in the server configuration file, *server_name*.conf. Tracing for the Reports Servlet is configured in the

servlet configuration file, `rwservlet.properties`. Tracing for individual jobs is specified from the runtime command line, via the TRACE command line arguments. See Report Trace for additional information about how tracing works with Oracle Reports.

- Trace information can only be generated when running a .rdf file. You cannot specify logging when running a .rep file.

A.4.112 TRACEOPTS

Table A–123 indicates which commands can use the TRACEOPTS keyword.

Table A–123   Commands that can use TRACEOPTS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	no	yes	yes	yes

Description   TRACEOPTS indicates the tracing information that you want to be logged in the trace file when you run the report.

Syntax   `TRACEOPTS={TRACE_ALL|TRACE_APP|TRACE_BRK|TRACE_ DBG|TRACE_DST|TRACE_ERR|TRACE_EXC|TRACE_INF|TRACE_LOG|TRACE_ PLS|TRACE_PRF|TRACE_SQL|TRACE_STA|TRACE_TMS|TRACE_WRN}`

Values   The following values apply:

- `TRACE_ALL` Log all possible trace information in the trace file.

- `TRACE_APP` Log trace information on all the report objects in the trace file.

- `TRACE_BRK` List all breakpoints in the trace file.

- `TRACE_DBG` Log debug information.

- `TRACE_DST` List distribution lists in the trace file. You can use this information to determine which section was sent to which destination.

- `TRACE_ERR` List error messages and warnings in the trace file.

- `TRACE_EXC` List Reports Server exceptions.

- `TRACE_INF` Dumps any information not covered by the other options into the trace file.

- `TRACE_LOG` Duplicate log information in your trace file. If you have specified a *log* element, in addition to a *trace* element, in your server configuration file, this

value will cause information that is sent to the log file to also be sent to the trace file.

- `TRACE_PLS` Log trace information on all the PL/SQL objects in the trace file.

- `TRACE_PRF` Log performance statistics in the trace file.

- `TRACE_SQL` Log trace information on all the SQL in the trace file.

- `TRACE_STA` Provide server and engine state information, such as initialize, ready, run, and shut-down.

- `TRACE_TMS` Enter a timestamp for each entry in the trace file.

- `TRACE_WRN` List server warning messages.

Default `TRACE_ALL`

Usage Notes

- In a runtime environment, or when you are monitoring Reports Services components, you can use all three `TRACE` keywords. But for server security, in a server environment you can use only `TRACEOPTS`. This is to prevent files from being written to the server.

- Tracing for the Reports Server is configured in the server configuration file, *server_name*`.conf`. Tracing for the Reports Servlet is configured in the servlet configuration file, `rwservlet.properties`. Tracing for individual jobs is specified from the runtime command line, via the `TRACExxx` command line arguments. See Report Trace for additional information about how tracing works with Oracle Reports.

- To use multiple options, list the options in parentheses. For example, `TRACE_OPTS=(TRACE_APP, TRACE_PRF)` means you want both `TRACE_APP` and `TRACE_PRF` applied.

- Trace information can only be generated when running a `.rdf` file. You cannot specify logging when running a `.rep` file.

- `TRACEOPTS` can be used when running JSP-based Web reports from the command line.

A.4.113 UNINSTALL

Table A–124 indicates which commands can use the `UNINSTALL` keyword.

Table A–124   Commands that can use UNINSTALL

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	no	no	no	yes

Description   Use UNINSTALL to remove an instance of the Reports Server from Microsoft Windows. This process removes the Reports Server service from the system. This keyword is ignored on UNIX.

Syntax   UNINSTALL *reports_server_name*

Values

reports_server_name The name of an existing Reports Server instance

Default   None

Usage Notes   If you use BATCH=YES with INSTALL, then none of the prompts and dialogs that normally display during the removal process will appear.

A.4.114 UPGRADE_PLSQL

Table A–121 indicates which commands can use the UPGRADE_PLSQL keyword.

Table A–125   Commands that can use UPGRADE_PLSQL

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	no	yes	no	no	no

Description   Use UPGRADE_PLSQL to specify whether to upgrade any PL/SQL code in the report to the latest version required by Oracle Reports Developer.

Syntax   UPGRADE_PLSQL=[YES|NO]

Values

- YES PL/SQL code will be upgraded automatically if necessary.

- NO PL/SQL code will not be updated.

Default   YES

A.4.115 URLPARAMETER

Table A–126 indicates which commands can use the URLPARAMETER keyword.

Table A–126   Commands that can use URLPARAMETER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	no	no	no	yes	yes	no

Description   Use URLPARAMETER to specify the URL that is to be fetched with the URL engine.

Syntax   URLPARAMETER=http://webserver_name/pagename.html

Values   Any valid URL.

Default   *None*

Usage Notes   This keyword is relevant when the *jobType* parameter of the *job* element in the Reports Server configuration file is rwurl, and a URL engine is in place.

A.4.116 USERID

Table A–127 indicates which commands can use the USERID keyword.

Table A–127   Commands that can use USERID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
yes	yes	yes	yes	yes	yes	no

Description   Use USERID only if you're not using Single Sign-On. Use USERID to specify your Oracle user name and password, with an optional database name for accessing a remote database. If the password is omitted, then a database logon form opens automatically before the user is allowed to run the report.

If you want users to log on to the database, then omit the password portion of the USERID keyword from the report request. If you want users to log on every time they run report requests, then use the Oracle Reports key mapping file, cgicmd.dat, to specify the runtime command, and include the %D option in the relevant key mapping entry.

> **Note:** For information on using the `cgicmd.dat` file, see
> **Running Report Requests**.

Syntax `userid=`*`username[/password`*`][@database]`

Values

username Username assigned by the database administrator.

password Password for the username.

database The name of the database you are accessing.

Default None

Usage Notes

- The logon definition cannot exceed 512 bytes in length.

- `USERID` can be used when running JSP-based Web reports from the command line.

A.4.117 WEBSERVER_DEBUG

Table A–128 indicates which commands can use the WEBSERVER_DEBUG keyword.

Table A–128   Commands that can use WEBSERVER_DEBUG

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	yes	no	no	no	no

Description Use `WEBSERVER_DEBUG` for JSP debugging. It creates the `stderr.log` and `stdout.log` files under the *docroot*/*port#* directory, and leaves temporary JSP files under *docroot*/*port#*/*default* and log files under *docroot*/*port#*/`log` for your inspection.

Syntax `WEBSERVER_DEBUG={YES|NO}`

Values

`YES` Create debugging files.

`NO` Do not create debugging files.

Default NO

Usage Notes

- Use this keyword only when you're running a job as a JSP.

- Relevant keywords include WEBSERVER_DEBUG, WEBSERVER_ DOCROOT, WEBSERVER_PORT.

A.4.118 **WEBSERVER_DOCROOT**

Table A–129 indicates which commands can use the WEBSERVER_DOCROOT keyword.

Table A–129   Commands that can use WEBSERVER_DOCROOT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	yes	no	no	no	no

Description   Use WEBSERVER_DOCROOT to set the Reports Builder document root directory. All files you reference in your JSP, such as images, HTML, and the like, should be relative to this directory. By setting the document root to your working directory, you avoid having to copy these files around.

Syntax   WEBSERVER_DOCROOT=*REPORTS_TMP/docroot*

For example:

```
WEBSERVER_DOCROOT=c:/temp/docroot
```

Values   The directory to the document root folder in your Oracle Reports temporary folder.

Default   None

Usage Notes

- Use this keyword only when you're running a job as a JSP.

- Relevant keywords include WEBSERVER_DEBUG, WEBSERVER_ DOCROOT, WEBSERVER_PORT.

A.4.119 **WEBSERVER_PORT**

Table A–130 indicates which commands can use the WEBSERVER_PORT keyword.

Table A–130   Commands that can use WEBSERVER_PORT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwcgi	rwserver
no	no	yes	no	no	no	no

Description   Use WEBSERVER_PORT to specify the port number an internal Web server listens to. You can specify a port number (e.g., 3002) or a range of port numbers (e.g., 3100-3200). If a single port number is specified, Oracle Reports tries to start the internal Web server listening on that port. If that port is in use, it tries to get the next available port. If a range of port numbers is specified, Oracle Reports tries to look for a free port in that range.

Syntax   WEBSERVER_PORT=*port num*

Values

port num   Any valid port number or range of port numbers.

Default

Port = 3000

Range of ports = 3000-3010.

Usage Notes

- Use this keyword only when you're running a job as a JSP.

- Relevant keywords include WEBSERVER_DEBUG, WEBSERVER_ DOCROOT, WEBSERVER_PORT.

B

Reports-Related Environment Variables

Environment variables are parameters that configure the environment that hosts Oracle Reports. The Oracle Application Server installer automatically defines default values for relevant environment variables. If you want something other than the default environment, you can edit the environment variable settings:

- For Windows, edit the environment variables through the Registry Editor (**Start > Run > Regedit**).

- For UNIX, edit the environment variables by revising and running the shell script that defines the initial default values (`reports.sh`). If you do this, be sure to keep a backup of the original, unaltered `reports.sh` file.

> **Note:** If you are upgrading to Oracle Reports 10*g* (9.0.4), the `reports.sh` file contains some required changes. If you have modified your current `reports.sh` file, save a backup and, after installing Oracle Reports 10*g* (9.0.4), merge your modifications into the `reports.sh` installed with 10*g* (9.0.4).

The information in this appendix is also documented in the *Reports Builder online help*, which you can access in two ways:

- From the Oracle Technology Network, (`http://otn.oracle.com/products/reports/`), click **Documentation** and navigate to the *Reports Builder Online Help* for the most recent, hosted online help.

- From Reports Builder, choose **Help > Help Contents**.

B.1 Environment Variables

This sections lists all the environment variables that you can set in Oracle Reports.

- For more information on all NLS environment variables, see the *Oracle Application Server 10g Globalization Guide* on the Oracle Technology Network, (`http://otn.oracle.com`).

- The environment variables shown with an asterisk (*) are supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

Note: The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

B.1.1 CA_GPREFS

Description This environment variable specifies the location of the global preferences file, CAGPREFS.ORA. Global preferences are shared among networked users. In addition to searching the directory specified by CA_GPREFS, products will also search the current directory for the CAGPREFS.ORA file.

The CAGPREFS.ORA file is automatically created by the Oracle Installer. To modify the global preference settings, use a text editor such as Notepad to manually edit this file. Global preferences set in the CAGPREFS.ORA file can be overridden by the local preference file, CAUPREFS.ORA, which is defined by CA_UPREFS.

Valid Values Any directory on any drive.

Default *ORACLE_HOME*

Example CA_GPREFS=C:\orawin90

B.1.2 CA_UPREFS

Description This environment variable specifies the location of the user preferences file, CAUPREFS.ORA. The CAUPREFS.ORA file maintains the preferences that you set via **Tools >Tools Options** within your products. In addition to searching the directory specified by CA_UPREFS, the product will also search the current directory for the CAUPREFS.ORA file.

Several Oracle products write their preference information to the CAUPREFS.ORA file. To manually modify the user preference settings, use a text editor such as Notepad to edit this file. User preferences set in the CAUPREFS.ORA file override global preferences set in the CAGPREFS.ORA file, which is defined by CA_GPREFS.

Valid Values   Any directory on any drive.

Default   *ORACLE_HOME*

Example   CA_UPREFS=C:\orawin90

B.1.3 DELIMITED_LINE_END

Description   This environment variable specifies whether to print the delimited character at the end of the line for delimited output.

Valid Values   YES|NO

Default   YES

Usage Notes
- Set this environment variable to NO to ensure that the delimited character is not printed at the end of the line.

B.1.4 DOC

Description   This environment variable specifies the location of the online documentation files, including online help.

Valid Values   Any directory on any drive.

Default   *ORACLE_HOME*\tools\doc

Example   DOC=C:\orawin90\tools\doc

B.1.5 INTERRUPT

Description   (DOS SQL*Net users only) This environment variable specifies the interrupt vector address that DOS Oracle products will use. If you use a DOS SQL*Net driver for Windows, check your `CONFIG.ORA` file located in the DOS Oracle home directory. If `INTERRUPT` is set in `CONFIG.ORA`, you must set it to the same value in registry. If this variable is not set in your DOS `CONFIG.ORA` files, you do not need to set it in the registry.

> **Note:**   Do not set `INTERRUPT` to 103 when using a third-party memory management product such as QEMM, 386MAX, CEMM, or EMM386. Doing so will result in unexpected behavior after you start SQLPME and SQL*Net.

Valid Values   Same value specified in your DOS CONFIG.ORA file.

Default   Not defined.

B.1.6 DEVELOPER_NLS_LANG

Description   This environment variable specifies the language for the Reports Builder. Chapter 18, "Implementing NLS and Bidirectional Support" contains additional detailed information about this environment variable, including a table of valid values.

B.1.7 NLS_CALENDAR

Description   This environment variable specifies the calendar system used.

B.1.8 NLS_CREDIT

Description   This environment variable specifies the string used to indicate a positive monetary value.

B.1.9 NLS_CURRENCY

Description   This environment variable specifies the local currency symbol.

B.1.10 NLS_DATE_FORMAT

Description   This environment variable specifies the default format used for dates.

B.1.11 NLS_DATE_LANGUAGE

Description   This environment variable specifies the default language used for dates.

B.1.12 NLS_DEBIT

Description   This environment variable specifies the string used to indicate a negative monetary value.

B.1.13 NLS_ISO_CURRENCY

Description   This environment variable specifies the ISO currency symbol.

B.1.14 NLS_LANG

Description   This environment variable specifies the language, settings used, including:

- The language used to display messages to the user, for example the 'Working...' message.

- The default format masks used for dates and numbers.

- The sorting sequence.

- The characters that make up the character set.

Chapter 18, "Implementing NLS and Bidirectional Support" contains additional detailed information about this environment variable, including a table of valid values.

Valid Values   See the NLS Reference Manual for a current list of available values, or see the *Oracle Application Server 10g Globalization Guide* on the Oracle Technology Network, (`http://otn.oracle.com`).

Default   `AMERICAN_AMERICA.WE8ISO8859P1`

Usage Notes

- To change locales, you must modify this environment variable, in addition to the REPORTS_RESOURCE environment variable.

- When you use features like OracleAS Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by OracleAS Reports Services may override the initial NLS_LANG setting. This change may in turn affect the behavior of the running report, such as bidirectional output in PDF. On UNIX platforms, you can work around this issue by setting the NLS_LANG environment variable explicitly in reports.sh. You can also use the new environment switching functionality to dynamically set the environment for Oracle Reports. Refer to Dynamic Environment Switching, for more information.

Example   Suppose you want your application to run in French. The application will be used in France and data will be displayed using the WE8ISO8895P1 character set. You would set NLS_LANG as follows:

```
NLS_LANG=French_France.WE8ISO8895P1
```

Now, suppose you still want your application to run in French, but this time it will be used in Switzerland. You would set NLS_LANG as follows:

```
NLS_LANG=French_Switzerland.WE8ISO8895P1
```

B.1.15 NLS_LIST_SEPARATOR

Description   This environment variable specifies the character used to separate items in a list.

B.1.16 NLS_MONETARY_CHARACTERS

Description   This environment variable specifies the decimal character and thousands separator for monetary values.

B.1.17 NLS_NUMERIC_CHARACTERS

Description   This environment variable specifies the decimal character and grouping separator for numeric values.

B.1.18 NLS_SORT

Description   This environment variable specifies the type of sort used for character data.

B.1.19 ORACLE_AFM

Description   This environment variable specifies the location of AFM files. TK90_ AFM is considered first, then ORACLE_AFM.

Valid Values   Any directory on any drive.

Default   Not defined.

Usage Notes
- If you do not specify values for either of these variables, Oracle Reports looks for AFM files in:

    ORACLE_HOME/guicommon9/tk90/admin/AFM

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.20 ORACLE_HOME

Description   This environment variable specifies the home directory in which Windows Oracle products are installed. This directory is the top directory in the Oracle directory hierarchy.

Valid Values   Any directory on any drive.

Default C:\orawin90

Usage Notes
- If you are using Reports Runtime (rwrun), the combined length of ORACLE_ HOME and ORACLE_PATH should not exceed 255 characters.

Example ORACLE_HOME=C:\orawin90

B.1.21 ORACLE_HPD

Description   This environment variable specifies the location of HPD files. TK90_ HPD is considered first, then ORACLE_HPD.

Valid Values   Any directory on any drive.

Default   Not defined.

Usage Notes
- If you do not specify values for either these variables, Oracle Reports looks for HPD files in:

  ORACLE_HOME/guicommon9/tk90/admin/HPD

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.22 ORACLE_PATH

Description   This environment variable specifies the search path for files referenced by Reports Runtime. Note that the directories specified by ORACLE_ PATH are searched after those specified by REPORTS_PATH.

ORACLE_PATH can specify multiple directories. Use a semi-colon (;) to separate directory names in a list of paths.

Valid Values   Any directory on any drive.

Default   Not defined.

Usage Notes
- If you are using Reports Runtime (rwrun), the combined length of ORACLE_ HOME and ORACLE_PATH should not exceed 255 characters.

Example   ORACLE_
PATH=C:\oracle\apps\forms;C:\oracle\apps\reports

B.1.23 ORACLE_PPD

Description   This environment variable specifies the location of PPD files. TK90_ PPD is considered first, then ORACLE_PPD.

Valid Values   Any directory on any drive.

Default   Not defined.

Usage notes

- If you do not specify values for either of these variables, Oracle Reports looks for TFM files in:

  ORACLE_HOME/guicommon9/tk90/admin/PPD

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.24 ORACLE_TFM

Description   This environment variable specifies the location of TFM files. TK90_ TFM is considered first, then ORACLE_TFM.

Valid Values   Any directory on any drive.

Default   Not defined.

Usage notes

- If you do not specify values for either TK90_TFM or ORACLE_TFM, Oracle Reports looks for TFM files in

  ORACLE_HOME/guicommon9/tk90/admin/TFM

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For information about printing on UNIX with Oracle Reports, refer to Printing on UNIX with Oracle Reports.

B.1.25 **ORAINFONAV_DOCPATH**

Description   This environment variable specifies the location of the table of contents and index for your online documentation.

Valid Values   Any directory on any drive.

Default   Not defined.

Example   ORAINFONAV_DOCPATH=C:\orawin90\oin

B.1.26 **PRINTER**

Description   This environment variable specifies the default printer's name.

Valid Values   Name of default printer

Default   Not defined.

Usage Notes

■   TK_PRINTER takes precedence over PRINTER, i.e., if both variables are set, TK_ PRINTER is considered first and PRINTER is considered only if TK_PRINTER does not specify a valid printer. If neither TK_PRINTER nor PRINTER is set to a valid printer, Oracle Reports uses the first entry in your uiprint.txt file. If REPORTS_NO_DUMMY_PRINTER is set, but the uiprint.txt file does not contain a valid entry, then screenprinter.ppd specified in uiscreenprint.txt is used.

> **Note:** REPORTS_NO_DUMMY_PRINTER is set by default and is required to be set at all times. If it is not set (as a result of being user-modified), then the REP-1800 error is raised.

See Also: ScreenPrinter

For more information on the PostScript printer driver, screenprinter.ppd.

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.27 REMOTE

Description   This environment variable specifies the default and remote SQL*Net driver to use when connecting via a local database. The parameter can include the default SQL*Net parameters (complete database string).

If a user logs on and specifies a connection with an explicit driver prefix matching the one specified in REMOTE, but specifies no SQL*Net parameters, the parameters specified in REMOTE are used. This parameter enables the DBA to define a "normal" network connection for which the SQL*Net user need not specify connection parameters. You can reset this parameter on the command line at any time.

> **Note:**   If you use a DOS SQL*Net driver for Windows, check to see whether the REMOTE parameter is set in your CONFIG.ORA file located in the DOS Oracle home directory. If REMOTE is set in CONFIG.ORA, you must set it to the same value in the registry.

Syntax   REMOTE netPrefix:databaseName

where

netPrefix is the network prefix for Named Pipes

databaseName is the name of the local database

Valid Values   Any valid SQL*Net driver prefix and parameters.

Default   Not defined.

Example   REMOTE=P:PIPER

where

P: is the network prefix for Named Pipes

PIPER is the database name.

B.1.28 REPORTS_ADD_HWMARGIN

Description   (Windows only) This environment variable specifies whether to include the printer hardware-based left margin. Oracle Reports has been changed to generate the report output relative to the top-left corner of the physical paper. In the past, the printer's printable area was used, causing inconsistencies in the location of the report output when moving across different printer models.

If required, you can revert to the previous behavior by setting the registry variable REPORTS_ADD_HWMARGIN to YES.

To set the REPORTS_ADD_HWMARGIN registry variable :

1. Edit the Windows registry using a registry editor (for example, regedit.exe).

> **Note:**   Before you edit the registry, back it up.

2. Navigate to the following key:

    HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOME*n*

    where *n* is the number of the *ORACLE_HOME* containing the installation.

3. Add a new String value named REPORTS_ADD_HWMARGIN and set the value to YES.

Valid Values   YES | NO

Default   NO

Usage Notes

■   When printing reports without hardware-based left margins on Windows, you must ensure that your report's layout contains enough margin spacing such that your data falls within the printable area. Margin fields in the Page Setup dialog box of Reports Builder have been disabled to ensure consistency with OracleAS Reports Services.

B.1.29 REPORTS_ARABIC_NUMERAL

Description   This environment variable specifies the numeric format for Arabic PDF output. Valid values for this environment are: ARABIC (Arabic numerals),

HINDI (Hindi numerals), or CONTEXT (Arabic or Hindi depending on the context). This environment variable is case insensitive.

Valid Values   ARABIC|HINDI|CONTEXT

Default   ARABIC

B.1.30 REPORTS_BIDI_ALGORITHM

Description   This environment variable switches the bidirectional (BiDi) layout algorithm for BiDi languages (for example, Arabic or Hebrew). This environment variable is case insensitive.

Valid Values   ORACLE|UNICODE

where

ORACLE Oracle Reports follows the Oracle BiDi algorithm. The Oracle algorithm is used in Oracle Application Server Forms Services.

UNICODE Oracle Reports follows the Unicode BiDi algorithm.

Refer to: http://unicode.org/unicode/reports/tr9/index.html, for more information on the Unicode BiDi algorithm.

Default   ORACLE

B.1.31 REPORTS_CGIDIAGBODYTAGS *

Description   This environment variable specifies the HTML attributes to add to the <BODY> tag in the rwcgi diagnostic and debugging output. For example, you can use this environment variable to set up text and background color or image.

Valid Values   Any valid HTML attributes for the <BODY> tag.

Default   Not defined.

Usage Notes
This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:** The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example REPORTS_CGIDIAGBODYTAGS="bgcolor="#CC3366""

B.1.32 REPORTS_CGIDIAGHEADTAGS *

Description This environment variable specifies the HTML tags to insert between the <HEAD> and </HEAD> tags in the rwcgi diagnostic and debugging output. For example, you can use this environment variable to set up <TITLE> or <META> tags.

Valid Values Any HTML tags that are valid between the <HEAD> and </HEAD> tags.

Default Not defined.

Usage Notes

- This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:** The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example REPORTS_CGIDIAGHEADTAGS="<title>Employee List</title>"

B.1.33 REPORTS_CGIHELP *

Description This environment variable specifies the URL and URI of the rwcgi help file that should display when rwcgi is invoked with the following empty request:

http://your_webserver/rwcgi?

Valid Values Any valid URL to a Web page or HTML file.

Default   A default help screen is displayed in your browser.

Usage Notes
- This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:**   The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Examples

To display the www.yahoo.com page in your browser:

REPORTS_CGIHELP=http://www.yahoo.com

To display an HTML file named myhelpfile.htm in your browser:

REPORTS_CGIHELP=http://your_webserver/myhelpfile.htm

B.1.34 REPORTS_CGIMAP *

Description   This environment variable specifies the fully qualified file name and location of the rwcgi map file, if map file configuration is used.

Valid Values   A valid path to the map file.

Default   *ORACLE_HOME*\reports\conf\cgicmd.dat

Usage Notes
- This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:**   The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example   REPORTS_CGIMAP=c:\ora9ias\reports\conf\cgicmd.dat

B.1.35 REPORTS_CGINODIAG *

Description This environment variable specifies whether to disable all debugging and diagnostic output, such as `help` and `showmap`, from `rwcgi`.

Valid Values `YES|NO`

Default `NO`

Usage Notes
- This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:** The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example The following request does not work when `REPORTS_CGINODIAG=YES`:

```
http://your_webserver/rwcgi/help?
```

B.1.36 REPORTS_CLASSPATH

Description This environment variable specifies the list of JAR files and directories for the Java Virtual Machine (JVM) when started by the Oracle Reports executables. You would typically add to this list when you need to include your own classes when designing reports (for example, when adding additional pluggable data sources (PDSs) or using the PL/SQL to Java bridge).

> **Caution:** Reports Builder will fail if the value of the `REPORTS_CLASSPATH` environment variable (registry) exceeds 511 characters. To work around this issue, you can use the `CLASSPATH` environment variable (system) to specify a value in excess of 511 characters.

Valid Values The default values are mandatory. If any of the entries are removed, the Oracle Reports executables may not behave correctly. Any additional

user-defined directory or JAR file that contains Java Classes may be appended to the path.

Default %ORACLE_HOME%\reports\jlib\rwbuilder.jar;%ORACLE_
HOME%\reports\jlib\rwrun.jar;%ORACLE_
HOME%\jlib\zrclient.jar;%ORACLE_
HOME%\j2ee\home\oc4j.jar;%ORACLE_HOME%\j2ee\home\lib\ojsp.jar

Usage Notes

- The default value for the environment variable is required for Oracle Reports executables to function correctly. Additional user classes may be appended, but the list must conform to the platform-specific Java CLASSPATH definition.

    Example

    ```
    REPORTS_CLASSPATH=%ORACLE_HOME%\reports\jlib\rwbuilder.jar;
    %ORACLE_HOME%\reports\jlib\rwrun.jar;%ORACLE_
    HOME%\jlib\zrclient.jar;%ORACLE_
    HOME%\j2ee\home\oc4j.jar;%ORACLE_
    HOME%\j2ee\home\lib\ojsp.jar
    ```

- Reports Builder will fail if the value of the REPORTS_CLASSPATH environment variable (registry) exceeds 511 characters. To workaround this issue, you can use the CLASSPATH environment variable (system) to specify a value in excess of 511 characters.

B.1.37 REPORTS_COOKIE_EXPIRE *

Description   This environment variable specifies the lifetime of a cookie within a given Reports Server session.

If Single Sign-On is not being used, then any user accessing a secured instance of the Reports Server is challenged to identify themselves by rwservlet via its own authentication mechanism (identical to the behavior of Oracle Reports6*i*). Because the HTTP 1.0 protocol is stateless (i.e., each call to the server is effectively independent of all others), users might need to authenticate themselves for each report request unless a cookie is maintained.

To allow users to authenticate themselves only once per session, rwservlet has its own client-side cookie, the authid cookie, in which it stores the required authentication information for the current session. Once the user is authenticated, an encrypted cookie is created in the browser to enable the user to submit multiple report jobs without re-authenticating for each request. The authid cookies are

terminated when the user closes their browser session, but you should not rely strictly on this method of terminating the cookie. You should limit the lifetime of the cookie within a given session using the REPORTS_COOKIE_EXPIRE environment variable. For example, a user might log on and then go to lunch, leaving the browser session open. To minimize the potential for a security breach in this situation, the administrator may define the REPORTS_COOKIE_EXPIRE environment variable on the Reports Server. When rwservlet receives a job request, 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 (e.g., 30 minutes), the cookie is rejected and the user is challenged to provide authentication information.

> **Note:** If you want to force users to authenticate themselves for a specific report, you can use the SHOWAUTH command line keyword. Alternatively, you can include a %S in the corresponding report entry in the key map file. This file is usually called cgicmd.dat and is located in *ORACLE_HOME*\reports\conf. %S forces users to enter their user name and password each time the report is called.

Valid Values   Any number of minutes.

Default   30

Usage Note
This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:** The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example   REPORTS_COOKIE_EXPIRE=30

B.1.38 REPORTS_DB_AUTH *

Description   This environment variable specifies the database authentication template used to log on to the database.

Valid Values   Any HTML file that contains special authentication actions. It is recommended that you keep the default.

Default   dbauth.htm

Usage Note

This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:**   The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example   REPORTS_DB_AUTH=dbauth.htm

B.1.39 REPORTS_DEFAULT_DISPLAY

Description   This environment variable specifies whether to implement the following features introduced with 10*g* (9.0.4):

- The elimination of the dependency on the DISPLAY environment variable (UNIX only)

- ScreenPrinter (screenprinter.ppd) for surface resolution for images and font information (UNIX only)

- Advanced Imaging Support (all platforms)

Valid Values   YES | NO

Default   YES

Usage Notes

- The Reports Server must be started in batch mode to suppress the UI.

- REPORTS_DEFAULT_DISPLAY=YES enables the enhanced imaging support introduced with the REPORTS_OUTPUTIMAGEFORMAT environment variable and OUTPUTIMAGEFORMAT command line keyword. The surface resolution can be controlled with the entry in the screenprinter.ppd file. If REPORTS_ DEFAULT_DISPLAY=NO, imaging support is limited to GIF format for PDF output, and HTML, HTMLCSS, and BMP for RTF output.

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

- On UNIX, REPORTS_DEFAULT_DISPLAY = YES overrides any value set for the DISPLAY environment variable.

- To revert to the dependency on DISPLAY and use screen fonts:

  - Set REPORTS_DEFAULT_DISPLAY=NO

  - Remove the screenprinter.ppd entry in the uiscreenprint.txt file.

B.1.40 REPORTS_DEFAULT_PIXEL_SIZE

Description   This environment variable specifies a pixel size that overrides the operating system's default pixel size when generating a report to HTML output. Normally, Oracle Reports takes its pixel size from the operating system. If you are working with older reports that rely upon a pixel size that is different from that of the operating system (for example, a pixel size of 80), you can use this variable to maintain the same behavior in your older reports.

Valid Values   Any value ranging from 72 through 200.

Default   Surface resolution determined by Oracle Reports.

Usage Notes
- For Windows, REPORTS_DEFAULT_PIXEL_SIZE is set in the registry. For UNIX, it is set from the command prompt or in a shell script.

- If REPORTS_DEFAULT_DISPLAY = YES (default), Oracle Reports still uses the value specified for REPORTS_DEFAULT_PIXEL_SIZE for HTML output. However, if a value is not explicitly set for REPORTS_DEFAULT_PIXEL_SIZE, the surface resolution is can be controlled with the entry in the screenprinter.ppd file, as described in Printing on UNIX with Oracle Reports.

B.1.41 REPORTS_ENCRYPTION_KEY *

Description   This environment variable specifies the encryption key used to encrypt the user name and password.

Valid Values   Any encryption key

Default   `reports9.0`

Usage Note

This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:** The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example   `REPORTS_ENCRYPTION_KEY=reports9.0`

B.1.42 REPORTS_GRAPH_IMAGE_DPI

Description   The `REPORTS_GRAPH_IMAGE_DPI` environment variable specifies a dots per inch (DPI) value for graphs output to a PDF file or a printer. The default value for this environment variable is set at 72 DPI to minimize the time taken to generate the report as well as to reduce the report file size.

If you specify a value higher than 72 DPI, you will see an improvement in the image resolution for graphs sent to a PDF file or a printer. However, this affects the time taken to generate the report output as well as the file size.

> **Note:** With the value of 250:
>
> - The time taken to generate a report with an Oracle Reports graph increases 5 to 6 times when compared to the time taken to generate the same report with the value set to 72dpi.
>
> - The PDF file size also increases 5 to 6 times.

Valid Values   72 through 300

Default   72

Usage Notes

- On Windows, use the registry to specify the value. On Unix/Linux, set the environment variable in `reports.sh`.

- When you set a higher DPI value, you may also need to change the JVM heap size value of `REPORTS_JVM_OPTIONS` to avoid the `Out Of Memory` error for the JVM

- To revert to the old behavior set the value of this environment variable to 72 DPI.

B.1.43 REPORTS_JVM_OPTIONS

Description   This environment variable specifies any JVM options that you want Reports Builder, Reports Runtime, or OracleAS Reports Services to consider when it starts its JVM. For example, you can use this environment variable to specify the starting heap size and maximum heap size for the JVM, additional classpath entries, and so on.

Valid Values   List of JVM options in the JVM command line syntax.

Default   `-Xmx256M`

Usage Notes

- The default value `-Xmx256M` specifies the JVM heap size of 256 to avoid the `Out Of Memory` error when running reports with large graphs or running big reports.

- In the server mode, you can also specify the JVM options in the `servername.conf` file in the `engine` attribute. If specified, the JVM options set in the server configuration file would take higher precedence to the `REPORTS_JVM_OPTIONS` environment variable.

B.1.44 REPORTS_NLS_XML_CHARSETS

Description   This environment variable provides an override option to enable you to define the character set encoding used when saving a report in XML format. This is only required where the required character set mapping for Reports `NLS_LANG` to XML IANA character sets do not produce the required results.

To enable your XML parser to understand the characters within the XML files, Oracle Reports does the following:

1. Adds an encoding attribute to the XML declaration based on the value in NLS_ CHARACTERSET, the character set part of the NLS_LANG variable.

2. Translates the value set as the NLS_LANG character set (e.g., JA16SJIS) to what is expected in the XML specification (e.g., Shift_JIS).

You can override this mapping by adding entries to the REPORTS_NLS_XML_ CHARSET.

Valid Values Set of mapping pairs separated by semicolons. The first value is the encoding that is being produced and the second mapped value is the value that should be used for these cases.

```
<old_name>=<new_name>[;<old_name>=<new_name>][;<old_name>=<new_name>]...
```

Default Not defined.

Usage notes This environment variable should not be required and is currently only used as a workaround to Bug 2224413:

```
NLS:NEED STABLE MAPPING FROM ORACLE'S NLS_CHARACTER_SET TO
IANA ENCODING
```

Example
```
ISO-8859-8=ISO-8859-8-1;CSEUCKR=EUC-KR;WINDOWS-949=EUC-KR;EUC-CN=GBK;WINDOWS-936=GBK
```

B.1.45 REPORTS_NO_DUMMY_PRINTER

Description This environment variable specifies whether the system's surface and fonts should be used instead of the printer's.

Valid Values TRUE|not set

Default TRUE

Usage Notes

- REPORTS_NO_DUMMY_PRINTER is set by default and is required to be set at all times. If it is not set (as a result of being user-modified), and there is no valid printer, error REP-1800 error is raised. Alternatively, you could use TK_ PRINT_STATUS when you have no valid printer. A valid printer response is

required by Oracle Reports to generate output, even if you are generating to a file. Beginning with Oracle Reports 10*g* (9.0.4), if the uiprint.txt file does not contain a valid entry (i.e., no valid printer is defined), but REPORTS_NO_DUMMY_PRINTER is set, Oracle Reports uses screenprinter.ppd specified in uiscreenprint.txt. You should unset this environment variable only if you do not want the screenprinter.ppd driver to be used by Oracle Reports when there is no valid printer.

> **See Also:** ScreenPrinter
>
> For more information on the PostScript printer driver, screenprinter.ppd.

- The limitation of this approach is that these reports might lose their formatting when viewed from another system if it is not identical to the system where the report was designed. Furthermore, when this report is printed, the formatting would not be correct because the fonts and their metrics differ.

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.46 REPORTS_NO_HTML_SPACE_REPLACE

Description

This environment variable specifies whether spaces should not be replaced with   in HTML or HTMLCSS output.

Oracle Reports will map HTML metadata characters in the data retrieved for a field to the appropriate encoding. That is, Oracle Reports automatically maps: < to < and " to ". In most cases, the browser produces the correct results and handles the spaces correctly. In some cases, the browser's handling of spaces does not produce the required output. This happens in such cases as where the user has padded the front of the data to produce indentation. Since the browser will treat multiple spaces as single space, the indentation is lost.

Valid Values   YES | not set

Default   not set

Usage Notes

- If the value is not set, all spaces are replaced by ` `. This could cause problems in your output where you want the browsers to handle line breaks on spaces. It will also increase the size of the generated HTML file.

- If a field's Contains HTML Tags property is set to YES, then no encoding will take place since Oracle Reports just passes the field's value through to the output.

Example

If `REPORTS_NO_HTML_SPACE_REPLACE` is set to `YES`, then the output for the sentence `[  Typical data output]` will be:

`[  Typical data output]`

and display as (ignoring preceding spaces):

`[Typical data output]`

Not setting the environment variable will cause the output to change to:

`[  Typical data output]`

and display as (maintaining preceding spaces):

`[  Typical data output]`

> **Note:** Brackets in the above example are used to show preceding spaces; they are not part of the sentence.

B.1.47 REPORTS_OUTPUTIMAGEFORMAT

Description   This environment variable specifies the default image format used in the report.

Valid Values   `GIF|JPEG|JPG|PNG|BMP`

Default   `JPEG`

Usage Notes

- You must ensure the format that you specify matches the output type. For example, BMP only works for RTF output. It will not work for HTML, HTMLCSS, or PDF output.

- This environment variable setting is overridden by the value of the OUTPUTIMAGEFORMAT command line keyword.

    Refer to Advanced Imaging Support for more information.

B.1.48 REPORTS_PATH

Description   This environment variable specifies the search path for files referenced by Reports Runtime. The directories specified by REPORTS_PATH are searched first, then those specified by ORACLE_PATH. This environment variable is used to locate reports and external objects that you use in your reports, such as PL/SQL libraries, external queries, and external boilerplate. It enables you to create reports that are easily portable across operating systems by preventing the need to hard-code directory paths.

Define REPORTS_PATH in the same fashion you define other environment variables on your operating system, keeping in mind such platform-specific rules as path length, etc.

In addition to directory paths, you can specify the keyword DB when you define REPORTS_PATH. This instructs Reports Builder to search the database to which you are currently connected.

Suppose you specified the following on a UNIX platform:

```
setenv REPORTS_PATH /home/tkostin/pay:/home/tkostin/receive:DB
```

Reports Builder will first search the directory /home/tkostin/pay. If it cannot find the file in question, it will search /home/tkostin/receive. If it still cannot find the file, it will search the database to which you connected when you began your Reports Builder session.

Valid Values   Any directory on any drive.

Default   %ORACLE_HOME%\REPORT\DEMO; %ORACLE_HOME%\REPORT\DEMO\BITMAP; %ORACLE_HOME%\REPORT\DEMO\REQFILES

Usage Note
- REPORTS_PATH is limited to 256 characters.
- If you specify a path for the sourceDir attribute of the engine element in the Reports Server configuration file (*server_name*.conf), the sourceDir value will override the values you set here.

Example

REPORTS_PATH=C:\oracle\apps\reports;C:\myfiles

B.1.49 REPORTS_RESOURCE

This environment variable specifies the location of the resource files required for reports. This path must include the NLS directory extension when specifying the location of the resource files.

Valid Values　Any directory on any drive.

Default　%ORACLE_HOME%\reports\res\US\

Usage Note

To change locales, you must modify this environment variable, in addition to NLS_ LANG.

Examples　For US files:

REPORTS_RESOURCE = %ORACLE_HOME%\reports\res\US\

For Japanese files:

REPORTS_RESOURCE = %ORACLE_HOME%\reports\res\JA\

B.1.50 REPORTS_SERVER *

Description　This environment variable specifies the default Reports Server for Web Cartridge or Web CGI requests. When this parameter is set, you can omit the SERVER command line keyword in report requests to process them using the default server, or you can include the SERVER command line keyword to override the default.

Valid Values　Any Reports Server service entry name.

Usage Note

This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:** The functionality of JavaServer Pages (JSPs) or servlets
> replaces support for CGI.

Example `REPORTS_SERVER=my_rep_server`

B.1.51 REPORTS_SPACE_BREAK

Description This environment variable specifies whether to consider white spaces
as a delimiter. Oracle Reports employs an algorithm to properly wrap a line, when a
word cannot fit in the same line. By default the word wrapping algorithm considers
white spaces as a delimiter.

Valid Values `YES | NO`

Default `YES`

Usage Note
Set this environment variable to `NO` only for Asian languages with multibyte
character sets, such as Chinese. This ensures that Oracle Reports does not consider
white spaces as delimiters and will enable appropriate word wrapping functionality
required by languages with multibyte character sets.

B.1.52 REPORTS_SSLPORT *

Description This environment variable specifies the port number when using SSL.

Valid Values Any valid port number.

Default 443

Usage Note
This environment variable is supported in Oracle Reports for backward
compatibility for Common Gateway Interface (CGI) reports, or when Single
Sign-On is not used.

> **Note:** The functionality of JavaServer Pages (JSPs) or servlets
> replaces support for CGI.

Example `REPORTS_SSLPORT=442`

B.1.53 REPORTS_SYS_AUTH

Description   This environment variable specifies the authentication template used to authenticate the username and password when users run report requests to a restricted Reports Server.

Valid Value   Any HTML file that contains special authentication actions. It is recommended that you keep the default.

Default   `sysauth.htm`

Usage Note
This environment variable is supported in Oracle Reports for backward compatibility for Common Gateway Interface (CGI) reports, or when Single Sign-On is not used.

> **Note:**   The functionality of JavaServer Pages (JSPs) or servlets replaces support for CGI.

Example   `REPORTS_SYS_AUTH=sysauth.htm`

B.1.54 REPORTS_TAGLIB_URI

Description   This environment variable specifies the location of the tag prefix used in the Web source of a JSP-based report. It defines the Reports URI of the tag library (TAGLIB) declarations of the .jsp file. This is typically:

```
<%@ taglib uri="/WEB-INF/lib/reports_tld.jar" prefix="rw" %>
```

When Oracle Reports finds a "uri" that matches the environment variable, it will use the corresponding "prefix" attribute to identify Oracle Reports tags within the .jsp file.

Valid Values   Any "uri" that references the Oracle Reports tag library.

Default   `/WEB-INF/lib/reports_tld.jar`

Usage Note

The default value is typically unchanged. It is the same for both Reports files in both JDeveloper and Reports. The "prefix" attribute can be changed to avoid naming conflicts independent of the "uri" attribute.

B.1.55 REPORTS_TMP

Description   This environment variable specifies the directory in which you wish to store Reports Builder temporary files. Reports Builder will use only one directory for this purpose; do not define more than one.

Define REPORTS_TMP in the same fashion you define other environment variables on your base operating system, keeping in mind such platform-specific rules as path length, and so on. If you don't define REPORTS_TMP, it will default to the current working directory.

Valid Values   Any directory on any drive.

Default   Not defined.

Example   REPORTS_TMP=C:\tmp

B.1.56 REPORTS_USEREXIT

Description   This environment specifies the DLLs for use by Reports Builder and its related executables (e.g., Reports Runtime). These DLLs are program modules created by you to be called by the Reports Builder executables.

REPORTS_USEREXIT can specify multiple DLLs. Use a backslash (\) to separate directories in a path, and a semicolon (;) to separate complete paths.

If this value is not explicitly set, Reports Builder looks for 60xtb32.dll according to the path variable of the system.

Valid Values   Any user exit dynamic link library (along with its absolute path).

Default   Not defined.

Example   REPORTS_
USEREXIT=C:\mydll.dll;d:\mynew.dll;e:\bin\speed.dll

B.1.57 RW

This environment variable specifies the reports-specific directory within the *ORACLE_HOME*.

Valid Values   A valid directory name.

Default

%ORACLE_HOME%\reports (Windows)

$ORACLE_HOME/reports (UNIX)

B.1.58 TK_PRINT

Description   (UNIX only) This environment variable specifies the print command to be executed on UNIX for Oracle Reports 6*i*. In Oracle Reports, TK_PRINT is *obsolete*; you can achieve the same results by using the printing script file: *ORACLE_HOME*/bin/rwlpr.sh. This script supports lp and lpr commands by default. If you use some other printing command for your machine, this file needs to be modified accordingly.

Valid Values   The PRINT command and all necessary keywords for your flavor of UNIX, including the following elements:

- %n is the printer name string.
- %c is the number of copies.

This string is much like a printf() format. If this environment variable is not set, Oracle Reports 6*i* uses the standard default value for the platform. Examples of default values on various platforms are as follows:

System V: lp -s -d'%n' -n%c

UnixWare: lp -s -Tpostscript -d'%n' -n%c

Solaris: lpr -P'%n' -#%c -s

Default   Not defined.

Usage Notes
- In most cases, the default print commands will meet your needs. We recommend that you only set this environment variable when you have a

specific need to alter the default value. For example, if you want duplexed output, you need to set TK_PRINT.

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.59 TK_PRINT_STATUS

Description   (UNIX only) This environment variable specifies the command executed to validate the printer. To make sure the printer is valid, this command is executed and its output is searched for the strings unknown, non-existent, or invalid. If one of these strings appears in the output, the printer is considered invalid and cannot be selected. Otherwise, the printer is accepted by Oracle Reports.

Valid Values   Should include %n for the printer name (see also TK_PRINT).

If this environment variable is not set, Oracle Reports uses the built-in default values:

System V: /usr/bin/lpstat -p'%n' 2>&1

Other: /usr/etc/lpc status '%n' 2>&1

Default   Not defined.

Usage Notes
- You should only use this environment variable in cases where the printer status command on your platform differs from the default values, or when you have no valid printer. If you have no valid printer, you can set TK_PRINT_STATUS=echo and specify a dummy entry in the uiprint.txt file. This workaround ensures that Oracle Reports gets a valid response when checking for a printer.

- If REPORTS_NO_DUMMY_PRINTER is set, but the uiprint.txt file does not contain a valid entry, then screenprinter.ppd specified in uiscreenprint.txt is used.

> **Note:**   REPORTS_NO_DUMMY_PRINTER is set by default and is required to be set at all times. If it is not set (as a result of being user-modified), error REP-1800 error is raised.

See Also: ScreenPrinter

For more information on the PostScript printer driver,
`screenprinter.ppd`.

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.60 TK_PRINTER

Description   (UNIX only) This environment variable specifies the default printer's name.

Valid Values   Name of default printer.

Default   Not defined.

Usage Notes

- `TK_PRINTER` takes precedence over PRINTER, i.e., if both variables are set, `TK_PRINTER` is considered first and `PRINTER` is considered only if `TK_PRINTER` does not specify a valid printer. If neither `TK_PRINTER` nor `PRINTER` is set to a valid printer, Oracle Reports uses the first entry in your `uiprint.txt` file. If `REPORTS_NO_DUMMY_PRINTER` is set, but the `uiprint.txt` file does not contain a valid entry, then `screenprinter.ppd` specified in `uiscreenprint.txt` is used.

  > **Note:** `REPORTS_NO_DUMMY_PRINTER` is set by default and is required to be set at all times. If it is not set (as a result of being user-modified), error `REP-1800` error is raised.

  See Also: ScreenPrinter

  For more information on the PostScript printer driver,
  `screenprinter.ppd`.

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.61 TK90_AFM

Description   This environment variable specifies the location of AFM files. `TK90_AFM` is considered first, then `ORACLE_AFM`.

Valid Values   Any directory on any drive.

Default   Not defined.

Usage Notes

- If you do not specify values for either of these variables, Oracle Reports looks for AFM files in:

   ORACLE_HOME/guicommon9/tk90/admin/AFM

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.62 TK90_HPD

Description   This environment variable specifies the location of HPD files. `TK90_HPD` is considered first, then `ORACLE_HPD`.

Valid Values   Any directory on any drive.

Default   Not defined.

Usage Notes

- If you do not specify values for either these variables, Oracle Reports looks for HPD files in:

   ORACLE_HOME/guicommon9/tk90/admin/HPD

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.63 TK90_PPD

Description This environment variable specifies the location of PPD files. TK90_
PPD is considered first, then ORACLE_PPD.

Valid Values Any directory on any drive.

Default Not defined.

Usage notes

- If you do not specify values for either of these variables, Oracle Reports looks for TFM files in:

  ORACLE_HOME/guicommon9/tk90/admin/PPD

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Printing on UNIX with Oracle Reports.

B.1.64 TK90_TFM

Description This environment variable specifies the location of TFM files. TK90_
TFM is considered first, then ORACLE_TFM.

Valid Values Any directory on any drive.

Default Not defined.

Usage notes

- If you do not specify values for either TK90_TFM or ORACLE_TFM, Oracle Reports looks for TFM files in

  ORACLE_HOME/guicommon9/tk90/admin/TFM

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For information about printing on UNIX with Oracle Reports, refer to Printing on UNIX with Oracle Reports.

B.1.65 USERNAME

Description   This environment variable specifies the variable part of a default logon account. See your database documentation for more information on setting USERNAME.

Valid Values   Any valid Oracle username (without the OPS$ prefix).

Default   Not defined.

Example   USERNAME=dsanvita

B.1.66 USER_NLS_LANG

Description   This environment variable specifies the language for the Oracle Reports Runtime component. Chapter 18, "Implementing NLS and Bidirectional Support" contains additional detailed information about this environment variable, including a table of valid values.

B.1.67 WIN_REMOTE_SESSIONS

Description   This environment variable specifies the number of connections to a remote server for which NETINIT7 should reserve memory.

Valid Values   0 - 33

Default   Not defined.

Usage Notes
- If you receive an error in the ORA-3100 to ORA-3199 range when you try to log on, there may not be enough real-mode memory to establish a communication buffer with an Oracle server. NETINIT7.EXE solves this problem by reserving memory for server connections.

- If you set WIN_REMOTE_SESSIONS to 0, NETINIT7 will not reserve any memory for remote server connections.

- Setting WIN_REMOTE_SESSIONS to the values 1 through 33 will reserve memory for the specified number of connections. Note that NETINIT7 reserves 12K of real-mode memory for each remote server connection.

- A suggested default value for WIN_REMOTE_SESSIONS is 5. Setting WIN_REMOTE_SESSIONS to 5 should create enough sessions to allow you to run several products.

- If you do not add the WIN_REMOTE_SESSIONS parameter, NETINIT7 will reserve memory for one remote server connection.

Configuring a Remote Session

To use NETINIT7**:**

1. Configure the number of sessions by adding the WIN_REMOTE_SESSIONS parameter.

2. Create the NETINIT7 icon in your Oracle program group. (For more information about creating icons, see your Microsoft Windows documentation.) Enter the following path in the Command Line field of the Windows Program Item Properties dialog box:

    ```
    %ORACLE_HOME%\bin\netinit7.exe
    ```

3. Start NETINIT7 by double-clicking the icon in the Oracle program group. Make sure that you start NETINIT7 before you start a product. If you want NETINIT7 to run automatically at Windows startup, include the NETINIT7 icon in the Windows Startup program group.

    NETINIT7 attempts to reserve real memory for the configured number of server connections. If NETINIT7 cannot allocate the required memory, it displays a warning message and indicates the actual number of sessions for which it has allocated memory.

    After allocating memory, NETINIT7 minimizes itself. Restoring NETINIT7 displays the total number of sessions configured and the total amount of memory used. Closing NETINIT7 releases all unused session memory. If an application is still using session memory, NETINIT7 notifies you that applications must log off before session memory can be freed.

    > **Note:** If you start a product before NETINIT7, it will allocate its own session memory, which is not recognized by NETINIT7. NETINIT7 attempts to allocate the configured session memory in addition to memory already allocated by the product you started.

B.1.68 XMMITR

Description   (DOS users only) This environment variable applies only to users of Oracle DOS protected mode executables. Check your CONFIG.ORA file, located in the DOS Oracle home directory. If the XMMITR parameter is set in CONFIG.ORA, you must set it to the same value in the registry. If this variable is not set in your DOS CONFIG.ORA files, you do not need to set it in the registry.

XMMITR (Extended Memory Manager Interrupt Vector) specifies the interrupt vector to be used for the extended memory manager/mode transition driver XMM/MTD interrupt services entry point. To set a different vector, the new interrupt vector is expressed as a decimal number. XMMITR should be used only if another program in your system needs to use interrupt 100 decimal, or 64 hexadecimal.

Valid Values   The same value specified in your DOS CONFIG.ORA file.

Default   Not defined.

C

Batch Registering Reports in OracleAS Portal

If you have a number of reports that you wish to register in OracleAS Portal, it is often preferable to register them as a group in a batch script rather than individually in the OracleAS Portal user interface. Likewise, if you have a large number of reports that you wish to unregister, a batch script is more efficient.

- Batch Registering Report Definition Files
- Batch Removing Report Packages
- PL/SQL Batch Registering Function

C.1 Batch Registering Report Definition Files

To batch register reports in OracleAS Portal, you need to perform the following steps:

1. Run rwconverter to Generate a SQL Script
2. Run the Script in SQL*Plus

C.1.1 Run rwconverter to Generate a SQL Script

To generate a SQL script that you can execute in SQL*Plus to register your reports, do the following:

1. From the operating system prompt (DOS or UNIX), enter the rwconverter command with the keywords to batch register the report definition files.

See Also: Appendix A, "Command Line Options"

For information on the `rwconverter` keywords.

Note: To successfully create a script file with the necessary load functions, you specify the DTYPE, STYPE, SOURCE, and DEST options. To create a functional package in OracleAS Portal, you will need to specify the P_SERVERS, P_PRIVILEGE, P_TYPES, P_FORMATS in addition to the options used to create the script file.

Following is an example `rwconverter` command line on Microsoft Windows:

```
rwconverter.exe dtype="register" stype="rdffile"
source="(security.rdf,earnings.rdf,acc_pay.rdf)" dest="(output.sql)"
p_owner="PORTAL_APP" p_servers="(repserver,acct_server)"
p_description="restricted report" p_privilege="(SCOTT,JABERS,ACCT)"
p_availability="production" p_types="(Cache,printer)"
p_formats="(HTMLCSS,PDF)" p_printers="(sales_printer,acct_printer)"
p_pformTemplate="public.finance_template"
p_trigger="Is begin IF UPPER(DESTYPE) = ''PRINTER'' AND
EMPNAME = ''SMITH'' THEN RETURN(TRUE); ELSE RETURN(FALSE); END IF; end;"
```

The above command line would generate a SQL script file named `output.sql` that contains the following:

```
SET SERVEROUTPUT ON

VAR STATUS NUMBER;

EXEC :STATUS := RWWWVREG.REGISTER_REPORT (P_NAME=>'Security',
P_OWNER=>'PORTAL_APP', P_SERVERS=>'repserver,acct_server',
P_FILENAME=>'security.rdf', P_DESCRIPTION=>'restricted report',
P_PRIVILEGE=>'SCOTT,JABERS,ACCT', P_AVAILABILITY=>'production'
P_TYPES=>'Cache,printer', P_FORMATS=>'HTMLCSS,PDF)',
P_PRINTERS=>'sales_printer,acct_printer
P_PFORMTEMPLATE=>'public.finance_template' P_PARAMETERS=>'(P_LASTNAME)
(P_SSN)', P_TRIGGER=>'Is begin IF UPPER(DESTYPE) = ''PRINTER'' AND
EMPNAME = ''SMITH''THEN RETURN(TRUE); ELSE RETURN(FALSE); END IF; end;');

EXEC :STATUS := RWWWVREG.REGISTER_REPORT (P_NAME=>'Earnings',
P_OWNER=>'PORTAL_APP', P_SERVERS=>'repserver,acct_server',
P_FILENAME=>'earnings.rdf', P_DESCRIPTION=>'restricted report',
P_PRIVILEGE=>'SCOTT,JABERS,ACCT', P_AVAILABILITY=>'production'
```

```
P_TYPES=>'Cache,printer)', P_FORMATS=>'HTMLCSS,PDF)',
P_PRINTERS=>'sales_printer,acct_printer',
P_PFORMTEMPLATE=>'public.finance_template',
P_TRIGGER='Is begin IF UPPER(DESTYPE) = ''PRINTER'' AND EMPNAME = ''JABERS''
THEN RETURN(TRUE); ELSE RETURN(FALSE); END IF; end;');

EXEC :STATUS := RWWWVREG.REGISTER_REPORT (P_NAME=>'Acc_pay',
P_OWNER=>'PORTAL_APP', P_SERVERS=>'repserver,acct_server',
P_FILENAME=>'acc_pay.rdf', P_DESCRIPTION=>'restricted report',
P_PRIVILEGE=>'SCOTT,JABERS,ACCT', P_AVAILABILITY=>'production'
P_TYPES=>'Cache,printer', P_FORMATS=>'HTMLCSS,PDF',
p_printers=>'sales_printer,acct_printer',
P_PFORMTEMPLATE=>'public.finance_template'
P_TRIGGER=>'Is begin IF UPPER(DESTYPE) = ''PRINTER'' AND
EMPNAME = ''JABERS''THEN RETURN(TRUE); ELSE RETURN(FALSE); END IF; end;');
```

For more information about the contents of this SQL script file, refer to PL/SQL Batch Registering Function.

2. Check the reports.log file, which is typically written to the current working directory, for errors that may have occurred during the conversion process. If the reports.log file was not generated, then no errors were encountered by rwconverter.

3. You can now optionally edit the system and user parameter values as desired. For example, the first RWWWVREG function in the sample script above generated an additional parameter called P_PARAMETERS. This occurred because the security.rdf file contains two user-defined parameters, P_LASTNAME and P_SSN:

```
P_PARAMETERS=>'(P_LASTNAME)(P_SSN)',
```

In this case, you can optionally define the default, low, and high values, or a list of values for each user parameter if you want to restrict the values the user may enter at runtime. Similarly, if you want to restrict system parameters, such as COPIES, to limit the number of copies a user can make, you do so by using the P_PARAMETERS parameter. The edited P_PARAMETERS keyword might look like the following:

```
P_PARAMETERS=>'(P_LASTNAME, LOV=LASTNAME_LOV)(P_SSN)(COPIES,
DEFAULT=1,LOW=1,HIGH=2)'
```

This revised code segment imposes the following restrictions on the report:

- The `P_LASTNAME` user parameter is limited to the values listed in the `LASTNAME_LOV` list of values.

- A user-supplied value for `P_SSN` is required.

- The default value of the `COPIES` system parameter is one and the number of printed copies must be in a range from 1 to 2.

4. Save and close the `output.sql` file.

C.1.2 Run the Script in SQL*Plus

To actually register your reports in OracleAS Portal, you must run the script generated for you by `rwconverter`:

1. Start SQL*Plus and connect to the OracleAS Portal schema that you want to own the packaged procedures.

2. From the SQL*Plus command prompt, execute the script you created with `rwconverter`:

   ```
   @ output.sql
   ```

   The script will execute and create packages in OracleAS Portal for each report listed in the script with the specified parameters.

3. Log in to OracleAS Portal as a user with `RW_ADMINISTRATOR` privileges.

4. Click the **Corporate Documents** tab.

5. Click **Builder**.

6. Click the **Administer** tab.

7. In the Oracle Reports Security portlet, click **Oracle Reports Security Settings**.

8. In the Reports Definition File Access portlet, enter the `P_NAME` of one of the reports you batch registered in your SQL script.

9. Click **Edit**. The Manage Component page is displayed.

10. Click **Edit** at the bottom of the page to edit the parameters of the report.

11. Review and edit the parameters as desired.

12. Click **OK**.

13. Click **Close**.

14. Repeat steps 8 through 13 for each report that you batch registered with your script.

C.2 Batch Removing Report Packages

To remove many reports from OracleAS Portal at once, do the following:

1. In a text editor, create a SQL script file (e.g., `rmv_rdfs.sql`) that contains one `RWWWVREG.DEREGISTER_REPORT` function call for each report definition file package that you want to remove. For example:

```
VAR STATUS NUMBER;
EXEC :STATUS := RWWWVREG.DEREGISTER_REPORT (P_NAME=>'Security');
EXEC :STATUS := RWWWVREG.DEREGISTER_REPORT (P_NAME=>'Earnings');
EXEC :STATUS := RWWWVREG.DEREGISTER_REPORT (P_NAME=>'Acc_pay');
```

> **Note:** `P_NAME` is the name of the report definition file package you want to remove from OracleAS Portal.

2. Start SQL*Plus and log in to the OracleAS Portal schema that owns the reports' packaged procedures.

3. From the SQL*Plus command prompt, execute the script you created in the first step:

```
@ rmv_rdfs.sql
```

The script will execute and remove the packages from OracleAS Portal for each report listed in the script.

> **Note:** This procedure will not remove the report definition files from the file system. It only unregisters the reports making them unavailable from OracleAS Portal. If you want to remove the files, you must delete them from the file system.

C.3 PL/SQL Batch Registering Function

The SQL script that `rwconverter` generates for you to batch register reports in Oracle Application Server consists mainly of calls to the `RWWWVREG.REGISTER_REPORT` function. The syntax of `RWWWVREG.REGISTER_REPORT` is as follows:

```
Function Rwwwvreg.register_report(
  p_owner varchar2,
  p_name varchar2,
  p_servers varchar2,
  p_filename varchar2,
  p_description varchar2,
  p_privileges varchar2,
  p_availability varchar2,
  p_types varchar2,
  p_formats varchar2,
  p_printers varchar2,
  p_pdformTemplate varchar2,
  p_parameters varchar2,
  p_trigger varchar2)
  return number;
    -- =0 : succeeded;
    -- !=0 : failed;
```

The table below describes each of the parameters taken by RWWWVREG.REGISTER_ REPORT.

Table C–1   RWWWVREG.REGISTER_REPORT *parameters*

Parameter	Description
P_OWNER	Is the DB Provider name. The default is the current Oracle Application Server DB Provider that you are connected to when you start the SQL*PLUS script.
	For example:
	P_OWNER=>'PORTAL_APP'
P_NAME	Is the name used to identify the report in OracleAS Portal.
	P_NAME corresponds to the Name field in the Create Report Definition File Access wizard.
	For example:
	P_NAME=>'Earnings'

Table C–1 `RWWWVREG.REGISTER_REPORT` *parameters*

Parameter	Description
P_SERVERS	Is the names of the Reports Servers on which the report definition files defined in the P_SERVERS parameter have access privileges. The list of Reports Servers is comma delimited.
	P_SERVERS corresponds to the Reports Servers field in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_SERVERS=>'repserver,acct'
	Note: The Reports Servers you list for P_SERVERS must already be registered in OracleAS Portal. For more information, refer to Chapter 11, "Deploying Reports in OracleAS Portal".
P_FILENAME	Is the name of the report definition file that is being registered.
	P_FILENAME corresponds to the Oracle Reports File Name in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_FILENAME=>'earnings.rdf'
P_DESCRIPTION	Is a description of the report.
	P_DESCRIPTION corresponds to the Description field in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_DESCRIPTION=>'restricted report'
P_PRIVILEGE	Is the users or roles given privileges to run the report definition file defined in P_FILENAME. This list is comma delimited.
	P_PRIVILEGE corresponds to the Grantee list on the Access tab of the Manage Component page for the report. Note that you must uncheck Inherit Privileges from Portal DB Provider in order to see the Grantee list.
	For example:
	P_PRIVILEGE=>'SCOTT,JABERS,PORTAL90'

Table C–1 RWWWVREG.REGISTER_REPORT *parameters*

Parameter	Description
P_AVAILABILITY	Is the name of the availability calendar that determines when the report definition file defined in the P_FILENAME parameter will be available for processing.
	P_AVAILABILITY corresponds to the Availability Calendar Name field in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_AVAILABILITY=>'production'
	Note: The availability calendar must already exist in OracleAS Portal. For more information about creating an availability calendar, see Chapter 11, "Deploying Reports in OracleAS Portal".
P_TYPES	Is the destination types to which the report definition file defined in the P_FILENAME parameter can be sent (e.g., cache, printer). This list is comma delimited.
	P_TYPES corresponds to the Types multiple select box in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_TYPES=>'CACHE,printer'
P_FORMATS	The destination formats to which the report definition file defined in the P_FILENAME parameter can be sent (e.g., HTML, PDF). This list is comma delimited.
	P_FORMATS corresponds to the Formats multiple select box in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_FORMATS=>'HTMLCSS,PDF'

Table C–1 RWWWVREG.REGISTER_REPORT *parameters*

Parameter	Description
P_PRINTERS	The printers to which the report definition file defined in the P_FILENAME parameter can print. This list is comma delimited.
	P_PRINTERS corresponds to the Printers multiple select box in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_PRINTERS=>'sales_printer,acct_printer'
	Note: The printers you list for P_PRINTERS must already be registered in OracleAS Portal. For more information, refer to Chapter 11, "Deploying Reports in OracleAS Portal".
P_PFORMTEMPLATE	Is the parameter form template that determines the page style of the Runtime Parameter Form.
	P_PFORMTEMPLATE corresponds to the Parameter Form Template field in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	For example:
	P_PFORMTEMPLATE=>'public.finance_template'

Table C–1 `RWWWVREG.REGISTER_REPORT` *parameters*

Parameter	Description
P_PARAMETERS	Is the user and system parameters' default, high, and low values, or list of values name.
	Note: The P_PARAMETERS parameter does not have a corresponding rwconverter option. Hence, if you want to batch import user parameter values, ranges, or lists of values, you must manually edit the SQL script generated by rwconverter.
	P_PARAMETERS corresponds to the (parameter) Name, LOV, Low Value, and High Value fields in the Create Report Definition File Access wizard and the Edit Report Definition File page.
	The default corresponds to the value set in the Runtime Parameter Form for the specified parameter.
	For example:
	`P_PARAMETERS=>'(P_LASTNAME, LOV=LASTNAME_LOV)` `(P_SSN)(COPIES, DEFAULT=1,LOW=1,HIGH=2)'`
	where:
	P_LASTNAME, P_SSN, and COPIES are parameter names.
	LOV is the name of the list of values.
	DEFAULT is the default value.
	LOW is the low value in a range of values.
	HIGH is the high value in a range of values.

Table C–1 `RWWWVREG.REGISTER_REPORT` *parameters*

Parameter	Description
P_TRIGGER	Is the validation trigger written in PL/SQL that returns a boolean statement (e.g., true (succeeded) or false (failed)).

P_TRIGGER corresponds to the text box in the Create Report Definition File Access wizard and the Edit Report Definition File page.

For example:

```
P_TRIGGER=>'Is begin
IF UPPER(DESTYPE) = ''PRINTER'' AND
  EMPNAME = ''SMITH''
THEN
  RETURN(TRUE);
ELSE
  RETURN(FALSE);
END IF;
end;'
``` |

Glossary

column

1. A vertical space in a database table that represents a particular domain of data. A column has a column name (e.g., ENAME) and a specific datatype (e.g., CHAR). For example, in a table of employee information, all of the employees' names would constitute one column. A record group column represents a database column.

2. A data model object created automatically for each column expression in a query's SELECT list, or created manually to perform summaries, formulas, or act as a placeholder.

3. The representation of an attribute of an entity.

data model

A relational model that defines what data should be fetched from the data source(s), what values should be computed, and how data should be ordered in a report. Reports Builder objects that define the data model are queries, groups, columns, parameters, and links.

Data Model view

One of the views of the Report Editor that displays a structural representation of the data in a report. The objects do not appear in the report output, but the structure determines the layout style, and the data objects provide the values that appear in the layout objects.

database

1. A set of dictionary tables and user tables that are treated as a unit.

2. (Oracle Express) A single file (possibly accompanied by extension files) that contains objects that organize, store, and manipulate data. In Express, examples of such objects are variables, dimensions, formulas, models, and programs.

data source

A source for data returned by a query, including database objects such as tables, views, synonyms, snapshots, and queries stored as views. OracleAS Reports Services allows you to access any data source.

The new pluggable data source (PDS) architecture replaces Oracle Open Client Adapter (OCA), and the Open Database Connectivity (ODBC) drivers are no longer supported in Oracle Reports 10g. However, Java Database Connectivity (JDBC) is one of the pluggable data sources available that can utilize the JDBC-ODBC bridge, allowing access to other data sources.

detail query

When defining a master/detail report, the detail query retrieves all related records for each record retrieved by the master, or parent, query.

dialog box

A partial screen or window that prompts you to enter information necessary to complete an operation.

disabled

An interface element state that means a menu item, button, and so on, cannot be used in the current context (i.e., it does not respond to keyboard or mouse input).

editor

See view.

enabled

An interface element state that means that a menu item, button, and so on, can be used in the current context (that is, it responds to keyboard or cursor/mouse input).

field

1. An interface element in which you enter, edit, or delete data.

2. A layout object that defines how the data for a specific query column appears.

foreign key

A value or column in one table that refers to a primary key in another table.

format mask

A setting that defines the appearance of the value of a field. For example, a format mask is used to specify the display of currency amounts and dates.

format trigger

A PL/SQL function that allows you to dynamically change the formatting attributes of an object.

formula column

A user-created column that gets its data from a PL/SQL function or expression, a SQL statement, or a combination of these.

frame

A layout object used to enclose other layout objects and control the formatting, frequency, and positioning of several objects simultaneously.

group

1. In Reports Builder, a data model object that is created automatically to contain all the columns selected by a query, or created by the user to modify the hierarchy of the data appearing in a report; it is used primarily for creating breaks in a report, as well as for resetting computations.

2. An object that is composed of several other objects.

HTML (HyperText Markup Language)

Acronym for 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)

Acronym for HyperText Transfer Protocol. The protocol used to carry Web traffic between a Web browser computer and the Web server being accessed.

hyperlink

A reference (link) from some point in one document to (some point in) another document or another place in the same document. A Web browser usually displays a hyperlink in some distinguishing way (in a different color, font or style). When users activate hyperlinks (by clicking on them with a mouse) the browser displays the target of the link.

icon

A graphic representation of a window or tool.

image

A bitmapped object that can be stored and loaded into an application. The client cannot modify an imported image.

intranet

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

Java

A computer language that supports programming for the Internet in the form of platform-independent "applets".

JAR

Acronym for Java ARchive. A file used for aggregating many files (Java class files, images, and so on) into one file.

J2EE

Acronym for Java 2 Platform, Enterprise Edition. An environment for developing and deploying enterprise applications in Java consisting of a set of services, application programming interfaces, and protocols that provide for developing multi-tiered, Web-based applications.

JSP (JavaServer Page)

JavaServer Page (JSP) technology is an extension to the Java Servlet technology from Sun Microsystems that provides a simple programming vehicle for displaying dynamic content on a Web page. JSP is a server-side technology. A JSP is an HTML page with embedded Java source code that is executed in the Web server or application server. The HTML provides the page layout that is returned to the Web browser, and the Java provides the business logic.

layout

See Paper Layout view.

margin

An optional report region that appears at the top and bottom of each logical page in a report section (Header, Main, or Trailer). The margin may include any layout object, but typically contains boilerplate and fields (for page numbers, page totals, grand totals, and current date and time).

object

1. An item that can be placed on the layout. The following are examples of objects: rectangle, line, ellipse, arc, polygon, polyline, rounded rectangle, freehand, chart, text, symbol, and text field.

2. In an Oracle database, an instance of an object type. An object can be a row in an object table, or the portion of a row contained in a column object in a relational table.

Object Navigator

A hierarchical browsing and editing interface that enables you to locate and manipulate application objects quickly and easily. Features include:

- A hierarchy represented by indentation and expandable nodes (top-level nodes show module types, database objects, and built-in packages), enabling tasks such as creating, editing, renaming, and deleting objects.

- A find field and icons, enabling forward and backward searches for any level of node or for an individual item in a node

- Icons in the horizontal toolbar replicating common File menu functions

Oracle Application Server (OracleAS)

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. OracleAS provides the only business-critical platform that offers easy database Web publishing and complete legacy integration while transitioning from traditional client-server to network application architectures.

Oracle Developer Suite

Combines leading Oracle application development and business intelligence tools into a single, integrated product. Built on Internet standards such as Java and XML, the suite provides a complete and highly productive development environment for building applications for Oracle Application Server and the Oracle database.

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.

You can refer to the directory specified by `ORACLE_HOME` in syntax:

On UNIX: `$ORACLE_HOME`

On Windows: `%ORACLE_HOME%`

OracleAS Portal

An HTML-based development tool for building scalable, secure, extensible HTML applications and Web sites. OracleAS Reports Services uses OracleAS Portal to control end user access to reports published on the Web by storing information about report requests, the secured server, and any OracleAS Reports Services printer used to print report output.

OracleAS Reports Services

See Reports Services.

Paper Design view

One of the views of the Report Editor that displays output for paper reports and allows you to make many commonly required, simple modifications to the layout, such as spacing, formatting fields, color, and editing text, without having to open the Paper Layout view.

Paper Layout view

One of the views of the Report Editor that displays the layout objects in a paper report and allows you to make many modifications to any layout object. All layout objects have properties that you can modify using the Property Inspector. The hierarchy of the layout objects is determined by the Data Model.

Paper Parameter Form view

Displays the layout of the Parameter Form that, at runtime, allows user input of parameter values in the Runtime Parameter Form.

PDF (Portable Document Format)

Acronym for Portable Document Format. A file format (native for Adobe Acrobat) for representing documents in a manner that is independent of the original application software, hardware, and operating system used to create the

documents. A PDF file can describe documents containing any combination of text, graphics, and images in a device-independent and resolution independent format.

PL/SQL

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

Property Inspector

A window that enables you to view, locate, and set the properties of the currently selected object(s) in the Object Navigator, Report Editor, and Template Editor. Every Reports Builder object (query, group, frame, parameter, etc.) has associated properties that can be viewed using the Property Inspector. The Property Inspector features:

- expandable and collapsible nodes
- in-place property editing
- search features
- multi-selection
- complex property dialogs
- the ability to invoke multiple instances of the Property Inspector

To get help on any property, click the property in the Property Inspector and press F1.

query

A SQL SELECT statement that specifies the data you wish to retrieve from one or more tables or views of a database.

RDF file

A file that contains a single report definition in binary format. .RDF files are used to both run and edit reports.

record

One row fetched by a SQL SELECT statement.

REP file

A file that contains a single report definition in binary format. .REP files are used solely to run reports; you cannot edit a .REP file.

repeating frame

A layout object used to display rows of data that are fetched for a group.

reports cache

A component of OracleAS Reports Services that stores completed jobs

Reports CGI (rwcgi)

An Oracle Reports executable, also known as the Common Gateway Interface (CGI) or Reports Web Cartridge, that translates and delivers information between either a Web Server or a J2EE Container (for example, OC4J) and the Reports Server, enabling you to run a report dynamically from your Web browser.

> **Note:** With Oracle Reports 10*g*, rwcgi is deprecated (maintained only for backward compatibility); instead, use JSPs or servlets (rwservlet).

Reports Client (rwclient)

An Oracle Reports executable that provides a command line interface to send a report to a remote Reports Server (rwserver).

Report Editor

The Reports Builder window that provides different views to help you handle the data objects and layout objects for Web and paper reports. The views are:

- Data Model view

- Paper Layout view

- Paper Design view

- Paper Parameter Form view

- Web source view

Reports Engine

A component of OracleAS Reports Services that fetches data from the data source, formats the report, send output to cache, and notifies the Reports Server that the job is ready.

Reports Builder (rwbuilder)

An Oracle Reports executable that starts Reports Builder to enable report developers to create and maintain report definitions.

Reports Queue Manager (rwrqm)

(Windows only) Maintains timestamp and status information about reports jobs managed by the Reports Server (`rwserver`).

Reports Runtime (rwrun)

An Oracle Reports executable that runs a report using the OracleAS Reports Services in-process server.

Reports Server (rwserver)

An Oracle Reports executable that enables running reports on a remote server. A component of OracleAS Reports Services that processes client requests, including user authentication, scheduling, caching, and report distribution. Use `rwclient` to send a report to the Reports Server.

Reports Services

The runtime environment for Reports Developer applications. OracleAS Reports Services executes, distributes, and publishes your reports for enterprise wide reporting. Using OracleAS Reports Services to deploy your reports results in gains of flexibility, time savings, and processing capacity.

Reports Servlet (rwservlet)

An Oracle Reports executable that translates and delivers information between either a Web Server or a J2EE Container (for example, OC4J) and the Reports Server, enabling you to run a report dynamically from your Web browser.

row

One set of field values in a table; for example, the fields representing one employee in the example table EMP.

Runtime Parameter Form

A screen or window appearing optionally at runtime in which a user can modify print options and parameters prior to report execution.

schema

A collection of related database objects, usually grouped by database user ID. Schema objects include tables, views, sequences, stored program units, synonyms, indexes, clusters, and database links.

SELECT statement

A SQL statement that specifies which rows and columns to fetch from one or more tables or views.

servlet

A Java application that runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing. Servlets provide an alternative to CGI (Common Gateway Interface) scripts. Because they are written in Java, servlets are portable between servers and operating systems.

The Reports Servlet and JSP are components of OracleAS Reports Services that process custom (JSP) report tags and deliver information between the Oracle HTTP Server and the Reports Server.

SQL

A standard interface for storing and retrieving information in a relational database. SQL is an acronym for Structured Query Language.

SQL file

A file that contains a query stored in text (e.g., ASCII or EBCDIC) format.

SQL script

A file containing SQL statements that you can run to perform database administration quickly and easily. Several SQL scripts are shipped with Oracle products.

SQL statement

A SQL instruction to Oracle. A SELECT statement is one type of SQL statement.

style sheet

HTML extensions that provide powerful formatting flexibility in HTML documents. To view an HTML document that takes advantage of style sheets, display it in a browser that supports style sheets.

table

A named collection of related information, stored in a relational database or server, in a two-dimensional grid that is made up of rows and columns.

tabular

A default layout displaying labels at the top of the page and rows of data underneath the labels.

template

A skeleton definition containing common style and standards, and may include graphics. A template provides a standard format to enable quick and easy development of professional standard look-and-feel reports.

Template Editor

A work area in which you can define objects and formatting properties for your templates. It is similar to the Paper Layout view of the Report Editor. You can create, delete, and modify objects (e.g., page numbers, text, and graphics) in the margin area. You cannot create and delete objects in the body area, but you can modify the properties of body objects in the Property Inspector.

tool

An iconic button used to create and manipulate objects in an application.

tool palette

A collection of tools represented by iconic buttons in the user interface that allow a report developer to perform tasks, such as drawing a rectangle in the Paper Layout view or creating a query in the Data Model view.

toolbar

A collection of iconic buttons that perform product commands. Usually aligned horizontally along the top, or vertically down the side of a window.

URL (Uniform Resource Locator)

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 OracleAS.

view

1. In Reports Builder, a work area in which you perform a specific set of tasks, such as defining a report data model, layout, or Parameter Form.

2. A virtual table whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.

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 (HTTP daemon) running at a Web site which sends out Web pages in response to HTTP requests from remote Web browsers.

Web source view

One of the views of the Report Editor that displays the HTML / JSP source for a report. You can use this view to add dynamic content to a Web page using the Report Block Wizard and the Graph Wizard. Experienced Java developers can edit the Web source directly in this view.

wizard

A step-by-step interface for commonly performed tasks. The wizards in Reports Builder are:

- Report Wizard: guides you through the steps to create a basic paper or Web report. Each page of the wizard asks you for information to help you create your initial report.

- Data Wizard: helps you helps you quickly define or modify a query for a multiquery data models.

- Graph Wizard: Adds variety of charts and graphs, including true 3-dimensional graphs. Implemented in Reports Builder with the Oracle BI graph bean.

- Report Block Wizard: enables you to add data to a static HTML page.

XML

Acronym for Extensible Markup Language. A metalanguage using SGML to define and structure data. Reports Builder supports XML output to enable Web publishing as well as electronic data exchange with third-party applications. You can also use XML to build report definitions that can be merged with other report definitions at runtime or run separately.

Index

C

K

S