Oracle® Fusion Middleware

Publishing Reports to the Web with Oracle Reports Services

12c (12.2.1.3.0)

E80071-01

August 2017

ORACLE®

Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services, 12c (12.2.1.3.0)

E80071-01

Primary Author: Arup Roy

Contributing Author: Igor Polyakov, Rajiv Malhotra, Ratheesh Pai, Vidya Viswanathan, Hariharan Srinivasan

Contents

7 Configuring Oracle Reports Services

10 Printing on UNIX with Oracle Reports

11 Using PDF in Oracle Reports

12 Font Model and Cross-Platform Deployment

13 Configuring Destinations for Oracle Reports Services

14 Configuring and Using the Pluggable Data sources

15 Securing Oracle Reports Services

16 Deploying Reports in Oracle Portal

19 Using the Oracle Reports Web Service

20 Creating Advanced Distributions

21 Using Event-Driven Publishing

Part VI Performance

24 Diagnosing and Tuning Oracle Reports

B Environment Variables

C Batch Registering Reports in Oracle Portal

D Troubleshooting Oracle Reports Services

E Reports Server and Bridge Diagnostic Utility

Preface

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

> **Note:** For the portable document format (PDF) version of this manual, when a URL breaks onto two lines, the full URL data is not sent to the browser when you click it. To get to the correct target of any URL included in the PDF, copy and paste the URL into your browser's address field. In the HTML version of this manual, you can click on a link to directly display its target in your browser.

Audience

This manual is intended for anyone who is interested in publishing reports with Oracle Reports Services. To configure Oracle Reports Services, it is useful 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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documentation

For more information, see the following documents:

- *Release Notes for Oracle Forms and Reports*
- *Installation Guide for Oracle Forms and Reports*
- *Oracle Reports Tutorial*
- *Oracle Reports User's Guide to Building Reports*
- *Oracle Reports online Help*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
`monospace italic`	Monospace italic type indicates variables or user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Part I

Getting Started

Part I contains information about Oracle Reports and Oracle Reports Services to get you ready to start publishing your reports:

1 Introduction

This chapter provides an overview of Oracle Reports, the reporting component of Oracle Fusion Middleware:

- Introduction to Oracle Reports

1.1 Introduction to Oracle Reports

Oracle Reports is Oracle's award-winning, high-fidelity enterprise reporting tool. It enables businesses to give immediate access to information to all levels within and outside of the organization in an unrivaled scalable and secure environment.

Using Oracle Reports, you can rapidly develop and deploy sophisticated Web and paper reports against any data source (including an Oracle database, JDBC, XML, and text files). Leveraging Java EE technologies such as JSP and XML, you can publish your reports in a variety of formats (including HTML, XML, PDF, Enhanced Spreadsheet, Spreadsheetdata, delimited text, delimiteddata, PostScript, and RTF) to any destination (including e-mail, Web browser, WebDav, FTP, Oracle Portal, and file system) in a scalable, efficient manner.

Oracle Reports includes the following component:

- Oracle Reports Builder

Oracle Reports Services is the focus of this manual. It executes, distributes, and publishes your reports for enterprise wide reporting. Using Oracle Reports Services to deploy your reports results in gains of flexibility, time savings, and processing capacity. It includes the following components:

- Oracle Reports Bridge
- Oracle Reports Client
- Oracle Reports Runtime
- Oracle Reports Servlet
- Oracle Reports Server

For more resources for information about Oracle Reports, refer to "Related Documentation" in the Preface.

1.1.1 Oracle Reports Builder

As its name implies, Oracle Reports Builder (`rwbuilder`) is the report-building component of Oracle Reports. Report developers use the Oracle Reports Builder design-time user interface to create and maintain report definitions, using:

- user-friendly wizards that guide you through the report design process
- pluggable data sources (PDSs), such as JDBC and XML, that provide access to data from any source for your reports
- a query builder with a graphical representation of the SQL statement to obtain report data
- default report templates and layout styles that can be customized if needed
- a live editor that enables you to modify paper report layouts in WYSIWYG mode
- the ability to add dynamic report output to an HTML page by embedding custom JSP tags within an HTML document
- an integrated graph builder to graphically represent report data
- the ability to generate code to customize how reports will run
- tools that dynamically generate Web pages based on your data
- standard report output formats such as HTML, HTMLCSS, XML, PDF, RTF, spreadsheet, PCL, PostScript, and ASCII
- client-side parameter validation using JavaScript
- the ability to execute dynamic SQL statements within PL/SQL procedures
- support for Oracle database objects
- event-based reporting (report execution based on database events)
- seamless integration of Oracle Reports with Oracle Portal for administering report security and publishing report output to portlets

---

Note: It is recommended that you check the TMP/TEMP values while installing. Since REPORTS_TMP can inherit these variable values, it is better to set them to a reliable directory, e.g. C:\TEMP before proceeding for the installation. However, if the installation has already been done in problem subdirectories, this can be resolved by:

- setting a new value like *C:\TEMP* in the registry
- setting a new value like *C:\TEMP* in opmn.xml

Problem subdirectories are the numbered subdirectories, e.g. *C:\Users\xxxxx\AppData\Local\Temp\3*, that are by default created by Microsoft (mstsc). Although these directories are used in TMP/TEMP environment variables, at some point of time they can be deleted by the Operating System.

For more information, refer to the *Oracle Reports online Help* (select **Help > Contents** in Oracle Reports Builder), and the *Oracle Reports User's Guide to Building Reports* manual.

1.1.2 Oracle Reports Bridge

Oracle Reports Bridge (`rwbridge`) provides functionality for discovering a Reports Server across farms. See Section 2.2.4.1.2, "Server Discovery Across Subnets".

1.1.3 Oracle Reports Client

Oracle Reports Client (`rwclient`) provides a command-line interface to send a report to a remote Reports Server (`rwserver`).

1.1.4 Oracle Reports Runtime

`rwrun` (Reports Runtime) runs a report by starting its own in-process server (not to be confused with the default in-process Reports Server), which runs in the same JVM as the `rwrun` process. The configuration file for this in-process server is `rwbuilder.conf` and trace files are saved in the `rep_`*`machinename`*`-rwbuilder` directory.

Note: It is recommended that you use `rwrun` for testing purposes only. Use `rwservlet` and `rwclient` in your production environment to take full advantage of the power of Oracle Reports Services.

1.1.5 Oracle Reports Servlet

Oracle Reports Servlet (`rwservlet`) is a component of Oracle Reports Services that translates and delivers information between either a Web Server or a Java EE Container (for example, Oracle WebLogic Server) and the Reports Server, enabling you to run a report dynamically from your Web browser.

1.1.6 Oracle Reports Server

Oracle Reports Server (`rwserver`) is a component of Oracle Fusion Middleware that provides reporting services to execute, distribute, and publish your reports for enterprise-wide reporting. This component processes client requests, including user authentication, scheduling, caching, and report distribution. Use Oracle Reports clients such as Oracle Reports Servlet (`rwservlet`), Reports JSP, and Oracle Reports Client (`rwclient`) to send a report to Oracle Reports Server (generally referred to as *Reports Server*).

2

Understanding the Oracle Reports Services Architecture

This chapter describes the architecture of Oracle Reports Services and its components. It also outlines considerations when setting up your Reports Server environment.

This chapter includes the following sections:

- Oracle Fusion Middleware Platform
- Oracle Reports Services
- Setting Up Your System
- Setting Up a High Availability Environment

2.1 Oracle Fusion Middleware Platform

Oracle Reports 12*c* Release (12.2.1.3) is integrated with Oracle Fusion Middleware and Oracle WebLogic Server, which results in simpler administration of complex topology and deployments. You can manage and monitor Oracle Reports components using either:

For complete overview and conceptual information about Oracle Fusion Middleware, refer to the following manuals:

- *Understanding Oracle Fusion Middleware*
- *Administering Oracle Fusion Middleware*

2.2 Oracle Reports Services

Oracle Reports Services is the reports publishing component of Oracle Fusion Middleware. 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 Oracle Reports Services to publish in both Web-based and non-Web-based environments.

Read this section to learn more about Oracle Reports Services:

- Section 2.2.1, "Overview"
- Section 2.2.2, "Oracle Reports Services Components"
- Section 2.2.3, "Oracle Reports Services Runtime Process"
- Section 2.2.4, "Oracle Reports Services Communication Architecture"

2.2.1 Overview

Oracle 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 Web Services. It enables you to run reports on both local and remote application servers and to implement a multitiered architecture for running your reports.

When used in conjunction with JSPs, Java servlets, or Web Services, Oracle Reports Services enables you to run reports on any platform from a Web browser using a standard URL syntax. For Oracle Reports Servlet (`rwservlet`) implementations, the in-process Reports Server is available for faster response and easier administration. The in-process Reports Server cuts down on the communication expense between processes and consequently shortens response times.

Oracle 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 7, "Configuring Oracle Reports Services").

Oracle Reports Services keeps track of all jobs submitted to the server, including jobs that are running, scheduled to run, finished, or failed. The `showjobs` Web command (through `rwservlet`), the Reports Queue Manager (Windows), and the Reports Queue Viewer (UNIX) 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 Oracle Reports Services, job information is persistent. This means that if the Reports Server is shut down then restarted, all jobs are recovered,[1] not just scheduled jobs.

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

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

- The client tier (a Web browser)
- The Web server tier
- The Oracle 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
- Oracle Reports Services tier

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

- The data tier, including databases and pluggable data sources

The client software could be:

- A `rwclient`
- A browser

The way you set up the 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. Refer to the *Planning an Installation of Oracle Fusion Middleware* for more information on sample topologies.

If you choose to have your Web server on multiple machines, you can cluster and load balance multiple Oracle Fusion Middleware instances for a highly available, fail-safe environment.

Note: Do not use "`server=clustername`" statements while calling jobs from an HA Reports Sever. For calling jobs from an HA Reports server, you must use a load balancer.

Refer to the *Planning an Installation of Oracle Fusion Middleware* for information on load balancing. Refer to *High Availability Guide* for information on high availability.

Oracle 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 21, "Using Event-Driven Publishing".

Oracle 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 20, "Creating Advanced Distributions".

2.2.2 Oracle Reports Services **Components**

Figure 2–1 Oracle Reports Services Components

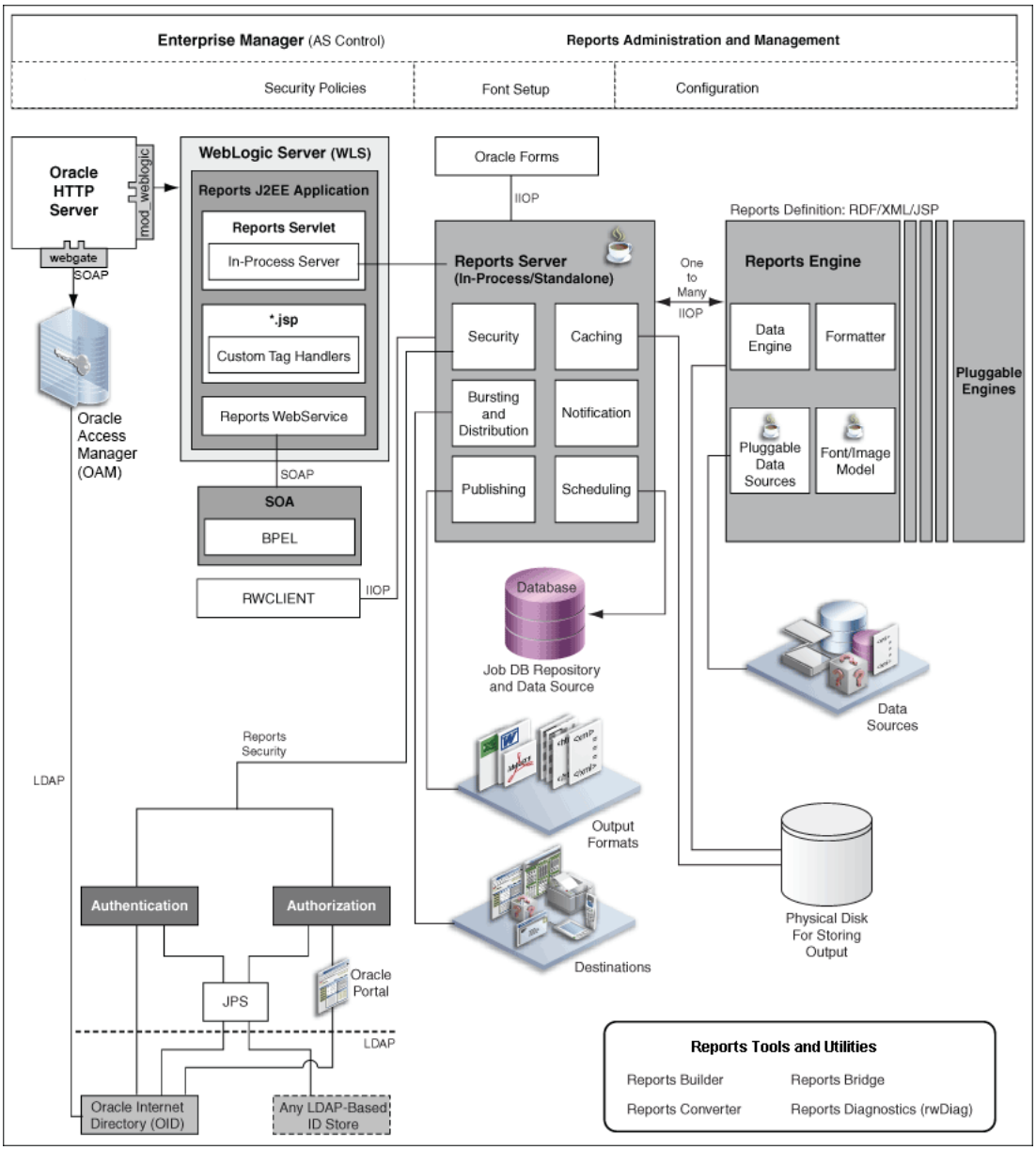


Figure 2–1 illustrates the components of a working Oracle Reports Services environment. This includes:

1. The **Oracle HTTP Server**, a Web server provided by Oracle Fusion Middleware. 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_weblogic**, used by the Oracle HTTP Server to redirect requests from servlets and JSPs to Oracle WebLogic Server. WebLogic Server provides a complete Java EE 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 Java EE environment. It is written entirely in Java and executes on the standard Java Development Kit (JDK) Virtual Machine (JVM).

3. The module **webgate**, used by the Oracle HTTP Server to connect to Oracle Access Manager (OAM). The Single Sign-On Server, in turn, uses Oracle Internet Directory.
4. **Oracle Reports Servlet** (rwservlet), a component of Oracle Reports Services that runs inside the Web server's servlet engine. Oracle Reports Servlet (rwservlet) translates and delivers information between HTTP and Reports Server. It uses Single Sign-On Server for authentication. Oracle Reports Servlet includes:
   - The in-process Reports 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 rwservlet or a Reports JSP.
5. 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.
6. The **Reports Server** (rwserver), which processes client requests, including ushering them through its various services, such as security (authentication and authorization, job scheduling, caching, publishing, and bursting and distribution (including distribution to custom—or pluggable—output destinations).

   The security service offers authentication based on the following methods:
   - Oracle Internet Directory
   - Java Platform Security (JPS) Oracle Internet Directory
   - Any LDAP-based ID Store using Java Platform Security (JPS)

   The security service offers authorization based on the following methods:
   - Java Platform Security (JPS) Oracle Internet Directory
   - XML file-based
   - Portal-based

   The bursting and distribution service enables the distribution of reports to multiple destinations, such as wireless, printer, FTP server, Portal, WebDAV, browser, and e-mail. The publishing service publishes Reports into multiple output formats, such as PDF, HTML, XML, and spreadsheet.

   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. Reports output can be stored in the Reports cache or on a physical disk.
7. The **Reports Server Cache**, which securely stores completed job outputs.
8. 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. The Reports engine also includes pluggable data sources, which are custom data sources like text, database, JDBC, OLAP, XML, and web services.
9. The **pluggable data sources**, a set of design-time and runtime Java APIs that provide openness to Reports by enabling data input from numerous sources

through the implementation of the PDS Java Interface. The PDS feature enables developers to leverage Reports' aggregation, summarization, formatting, and scheduling capabilities not only on data that is accessed through SQL, but also on data that is available elsewhere.

10. 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. Oracle 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 destinations such as e-mail, Oracle Portal, and WebDAV.

    In addition to the Reports Services components, Figure 2–1 depicts the following:

    - Integration of Oracle Reports with Service Oriented Architecture (SOA) through the web services module of the Reports J2EE application.
    - Communication between Oracle Forms and Oracle Reports.

Additionally, the **Oracle Reports Bridge** provides functionality for discovering a Reports Server across Farms. The Oracle Reports Bridge acts as a gateway for packets that are broadcast by Reports Server/Reports Client across Farms. The Oracle Reports Bridge mechanism is not shown in Figure 2–1; see Section 2.2.4.1.2, "Server Discovery Across Subnets".

2.2.3 Oracle Reports Services **Runtime Process**

The various components of Oracle Reports Services contribute to the process of running a report as follows:

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 or `rwservlet`, both associated with the Oracle HTTP Server. The requests go to `mod_weblogic`. (For jobs that run as JSPs, `mod_weblogic` uses OJSP to translate the JSP into a servlet.)

      The URL may contain runtime parameters or a keyword that refers to a runtime parameter configuration section within the `cgicmd.dat` key map file (see Section 18.14, "Using a Key Map File"), 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` component translates and delivers information between either a Web server or a Java EE Container (for example, Oracle WebLogic Server) and the Reports Server:

    Server requests from Reports JSP or `rwservlet` can be run by the in-process Reports Server or as a standalone Reports Server process (recommended), whichever is specified in the Oracle Reports Servlet (`rwservlet`) configuration file (`DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration//rwservlet.properties`). An in-process Reports Server requires less maintenance than a standalone Reports Server because, unlike the standalone Reports Server, it starts automatically in response to

requests from the client. An in-process Reports Server cuts down on the communication between processes. A standalone server, on other hand, provides better control outside the `rwservlet` process with the ability to separate out server process from the WebLogic Server instance. For information about specifying an in-process Reports Server and default naming, see Section 7.3, "Oracle Reports Servlet Configuration File".

3. The Reports Server processes the request:

   If the request includes a `TOLERANCE` option, 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` option. `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` option 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). See Section A.8.22, "TOLERANCE" and Section A.6.6, "EXPIRATION".

   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:

   a. If Oracle Reports Servlet (`rwservlet`) is SSO-enabled, it checks for authentication. A secure Reports Server then authorizes the user using Oracle Internet Directory. If Oracle Reports Servlet (`rwservlet`) is not SSO-enabled, a secure Reports Server authorizes and authenticates the user.

   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 *`rwserver`*`.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 21, "Using Event-Driven Publishing".

For information about running reports with Oracle BPEL Process Manager, refer to Section 7.10, "Configuring Oracle Reports to Communicate with Oracle BPEL Process Manager".

2.2.4 Oracle Reports Services Communication Architecture

Oracle Reports replaces the use of Borland's VisiBroker with Sun Microsystems' industry-standard Java Developer's Kit Object Request Broker (JDK ORB), providing support for Reports Server requests from clients across subnets, and using the broadcast mechanism for dynamic Reports Server discovery both within a subnet and across subnets.

With Oracle Reports 12*c* Release (12.2.1.3), you can use the built-in broadcast mechanism, available out-of-the-box, for dynamic discovery of Reports Servers. You can also choose to use the Common Object Service (COS) naming service orbd, provided by Sun Microsystem's JDK ORB, for Reports Server discovery.

Note: It is recommended that you use the built-in broadcast mechanism for dynamic discovery of Reports Servers. Use the Common Object Service (COS) naming service for Reports Server discovery only when the built-in broadcast mechanism is not suitable for your environment, as in the following scenarios:

- You plan to install Oracle Reports on a machine that is connected to a network using VPN.
- You want to avoid broadcast traffic on your network.

`namingService` is not supported on Reports High Availability.

This section discusses the two methods of Reports Server discovery

- Server Discovery Using the Broadcast Mechanism
- Server Discovery Using the COS Naming Service

Note: Oracle Reports contains `rwdiag` executable to provide diagnosis for the JDK ORB implementation. Using rwdiag, you can replace the functionality of osfind available in the prior VisiBroker implementation, providing information about which ORB applications are running and options for logging ORB-related network traffic

---

2.2.4.1 Server Discovery Using the Broadcast Mechanism

With the broadcast mechanism, Reports Server discovery can occur within a subnet or across subnets:

- Server Discovery within a Subnet
- Server Discovery Across Subnets

Note: The Oracle Reports built-in broadcast mechanism requires the host machine to be inside a network. Thus, following are two scenarios in which the broadcast mechanism may not work, including the solutions for each scenario:

1. If the host machine is not in a network (that is, it is a standalone machine).

   Solution for Windows platform: Install the MS loopback adapter, as described on the Microsoft Web site (http:\\microsoft.support.com). Then, specify an IP address for your machine (either XP or Windows 2000), as follows:

   1. On your desktop, right-click **My Network Places**, and choose **Properties**.
   2. Right-click the MS loopback adapter, and choose **Properties**.
   3. In the Properties dialog box, select **Internet Protocol (TCP/IP)**, and click **Properties**.
   4. Select **Use the following IP address**, and enter a valid IP address. The subnet mask field will automatically populate. Do not use the local host IP (that is, 127.0.0.1). For example, enter: 198.162.1.1.
   5. Click **OK**, and follow the instructions displayed.

   Solution for UNIX platform: Configure the discovery mechanism to disable the built-in broadcast mechanism and enable the COS naming service instead, by replacing the `multicast` element with the `namingService` element in the network configuration file (`rwnetwork.conf`).

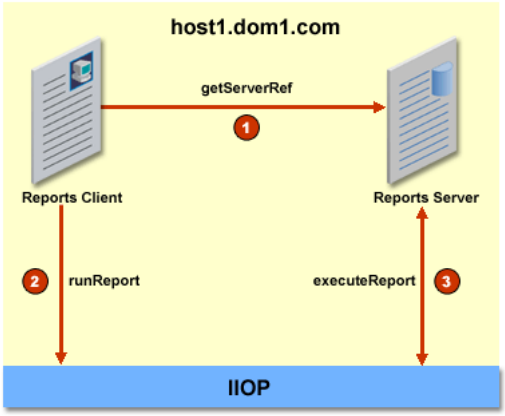
2. If the host machine is connected to a network through VPN.

   Solution for both Windows and UNIX platform: Configure the discovery mechanism to disable the built-in broadcast mechanism and enable the COS naming service instead, by replacing the `multicast` element with the `namingService` element in the network configuration file (`rwnetwork.conf`), as described in Section 2.2.4.1, "Server Discovery Using the Broadcast Mechanism".

2.2.4.1.1 Server Discovery within a Subnet

Within a subnet, the client broadcasts a packet with the name of the Reports Server to which it wants to connect. A Reports Server with that name will respond if it exists in the network. The client then connects to the Reports Server to run the report request.

Figure 2–2 Server Discovery within a Subnet



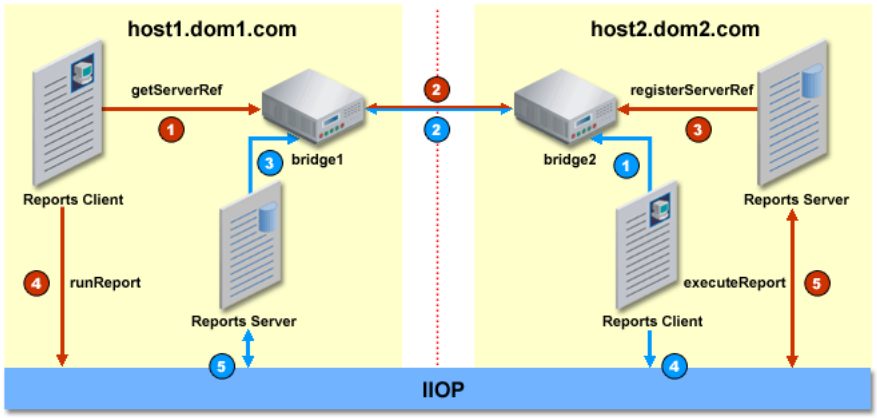
Using the example of running a report request with `server=rep_server`, the following numbered steps map to the numbers in Figure 2–2:

1. `getServerRef`
   - The Oracle Reports client (for example, `rwclient`, `rwservlet`, or `rwrqm`) broadcasts a packet containing the name of the server to which it wants to connect. In this case, the packet contains the name `rep_server`.
   - The server with name `rep_server` in the network responds back with its Interoperable Object Reference (IOR).
   - The client converts the IOR to an object reference.
2. `runReport`
   - The Oracle Reports client sends a request to the remote object reference to run the report (IIOP call).
3. `executeReport`
   - Reports Server executes the report and returns either report or status.

2.2.4.1.2 Server Discovery Across Subnets

Oracle Reports provides the Oracle Reports Bridge mechanism for connecting two or more non-secured subnets. An Oracle Reports Bridge running in one subnet will contact Oracle Reports Bridge running in another subnet to obtain Reports Server references. For configuration details, refer to Oracle Reports Bridge Configuration Elements.

Figure 2–3 Server Discovery Across Subnets



Using the example of running a report request with `server=rep_server`, the following numbered steps map to the numbers in Figure 2–3:

1. `getServerRef`
   - The Oracle Reports client (for example, `rwclient`, `rwservlet`, or `rwrqm`) broadcasts a packet containing the name of the server to which it wants to connect. In this case, the packet contains the name `rep_server`.
2. `bridge1` intercepts the packet and passes it to `bridge2`.
3. `registerServerRef`
   - `bridge2` broadcasts the packet in `dom2`, and `rep_server` in `dom2` responds back with the Interoperable Object Reference (IOR).
   - `bridge2` passes the IOR back to `bridge1`, and `bridge1` passes it to the client using the broadcast mechanism.
   - The Oracle Reports client converts the IOR to an object reference.
4. `runReport`
   - The Oracle Reports client sends a request to the remote object reference to run the report (IIOP call).
5. `executeReport`
   - Reports Server executes the report and returns either report or status.

> **Note:** Numbers in blue color in Figure 2–3 are shown for the case where the Oracle Reports client is in `dom2` and Reports Server is in `dom1`.

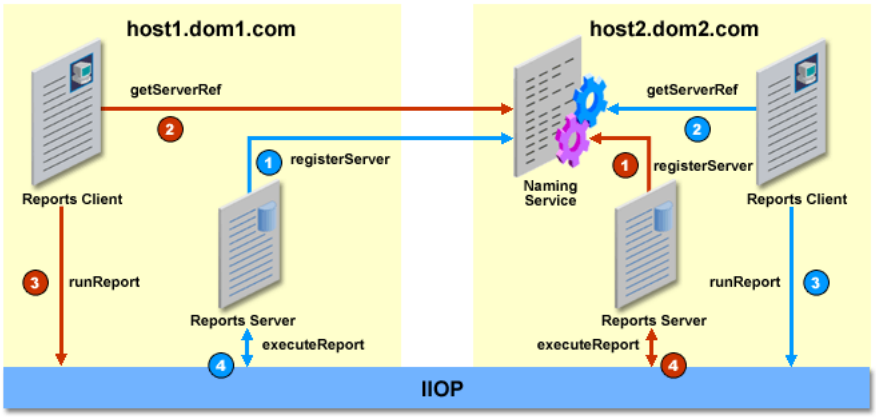
2.2.4.2 Server Discovery Using the COS Naming Service

Alternatively, you can use the JDK-provided Common Object Service (COS) naming service to access a Reports Server in the same subnet, as well as across a non-secured subnet. To configure the naming service, refer to Section 7.5.1, "Network Configuration Elements".

Note: It is recommended that you use the built-in broadcast mechanism for dynamic discovery of Reports Servers. Use the Common Object Service (COS) naming service for Reports Server discovery only when the built-in broadcast mechanism is not suitable for your environment, as in the following scenarios:

- You plan to install OracleAS Reports Services on a machine that is connected to a network using VPN.
- You want to avoid broadcast traffic on your network.

Figure 2–4 Server Discovery Using the COS Naming Service



Using the example of running a report request with server=rep_server, the following numbered steps map to the numbers in Figure 2–4

1. `registerServer`
   - When rep_server is started, it registers itself with the naming service, which must be up and running for the server to start. Now a request is run with `server=rep_server`.
2. `getServerRef`
   - The Oracle Reports client contacts the naming service and requests a reference to `server rep_server`.
   - The naming service returns the reference to server rep_server.
3. `runReport`
   - The Oracle Reports client sends a request to the remote object reference to run the report (IIOP call).
4. `executeReport`
   - Reports Server executes the report and returns either report or status.

Note: Numbers in blue color in Figure 2–4 are shown for the case where the Oracle Reports client is in `dom2` and Reports Server is in `dom1`.

2.3 Setting Up Your System

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

The following subsections discuss some of the decisions involved in:

- Choosing the Types of Requests You Will Service
- Choosing Oracle Reports Servlet, JSP, or Web Services
- Choosing Single or Multiple-Machine Configurations

2.3.1 Choosing the Types of Requests You Will Service

Oracle 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 Oracle Reports Servlet (`rwservlet`) or a JSP with your Web server. One or the other of these components must be present on the Web server to enable communications between it and Oracle Reports Services and to enable the processing of report requests from Web clients.

> **Note:** For more information, refer to Section 2.3.2, "Choosing Oracle Reports Servlet, JSP, or Web Services".

In non-Web cases, you can send job requests using the `rwclient` component, 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 Oracle Reports Servlet (`rwservlet`) or JSP 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 client software could be:

  - An Oracle Developer Suite installation, which can be used to submit reports requests through `rwclient` or `rwrun`.
  - A utility such as the Oracle Reports Java EE Thin Client, which is available on the Oracle Technology Network (OTN). Note that this utility is supported only through Discussion Forums on OTN.

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

2.3.2 Choosing Oracle Reports Servlet, JSP, or Web Services

To use Oracle Reports Services in a Web environment, you must use Oracle Reports Servlet (`rwservlet`), Reports Web Services, or Reports JSP.

Between Oracle Reports Servlet (`rwservlet`) and Reports JSP there are additional considerations. A JSP-only implementation means that you can publish a layout that is optimized for Web delivery (that is, the Oracle Reports Web Layout). Oracle Reports Servlet (`rwservlet`) enables you to include paper layouts in your report publishing solution and fully leverage the distribution features of Oracle Reports Services.

Using Oracle Reports Servlet (`rwservlet`) 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 option `report=`*`myreport`*`.jsp`. In this case, report output is created based on the paper layout.

For more information on running reports, see Chapter 18, "Running Report Requests".

2.3.3 Choosing Single or Multiple-Machine Configurations

You can place Oracle 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 Oracle 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 Reports Server (available only with Oracle Reports Servlet (`rwservlet`) implementations) you can further amplify the performance advantages of a single machine. The in-process Reports Server speeds up processing time by allowing for faster and more efficient communication between Oracle Reports Services components. We recommend that you use the in-process Reports Server unless you will not use Oracle Reports Servlet (`rwservlet`) 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 Reports Server will aid performance by speeding communication between Oracle 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 enables you to implement load balancing on the Web server.

By using the environment switching feature it is possible to spawn Reports Engines with different environment settings, including language, in the same Reports Server. Refer to Section 7.2.2, "Dynamic Environment Switching" for more information.

2.4 Setting Up a High Availability Environment

This section discusses the following topics pertinent to establishing a high availability environment:

2.4.1 Maintaining High Availability

Oracle Fusion Middleware consists of many components that can be deployed in distributed topologies. The underlying paradigm used to enable high availability for Oracle Fusion Middleware is clustering, which unites various Oracle Fusion Middleware components in certain permutations to offer scalable and unified functionality, and redundancy should any of the individual components fail.

Note: For more information configuring high availability for Reports in Oracle Fusion Middleware, see *High Availability Guide*.

2.4.2 Configuring the rwservlet.properties file

Perform the following steps to configure the rwservlet.properties file:

1. Make sure that the in-process server has a unique name. You can check the server sub-element of the rwservlet.properties to verify the unique server name.
2. Configure cluster in the rwservlet.properties file.

   ```
   <cluster clustername="ha_cluster" clusternodes="ha_server2"/>
   ```

   `cluster name` should be same in all rwservlet.properties files. `Cluster nodes` should include in-process server names which are part of the cluster except the current one. More than one server should be separated by a colon.
3. Save the servlet configuration file.

Alternatively, you can configure a cluster through Oracle Enterprise Manager:

1. Navigate to the Reports Application Home Page in Enterprise Manager.
2. From the **Reports** menu, select **System MBean Browser**.

   The **System MBean Browser** page is displayed.
3. From the left content pane, expand **Application Defined MBeans** > **oracle.reportsApp.config** > **Server: WLS_REPORTS** > **Application: reports** > **ReportsApp**.
4. Select **rwservlet** from the list.

   The **Application Defined MBeans: Report sApp:rwservlet** page is displayed.
5. Click the **Operations** tab.
6. Select **addCluster**.

   The **Operation:addCluster** page is displayed.
7. Enter valid values in the **Cluster Name** and **Node** fields.
8. Click **Invoke**.

For more information about the rwservlet.properties file, refer to Section 7.3.1.1, "rwservlet"

2.4.3 Configuring Reports Server for High Availability

To configure Reports Server for high availability by using the database as the job repository, perform the following steps for each instance:

1. Configure the database job repository for in-process Reports Servers in all instances in the `rwserver.conf` file of all Reports Servers.

   Note that in 12*c* Release (12.2.1.3) the server configuration file must correspond to the `rwserverconf.xsd` file (refer to Section 7.2.1, "Reports Server Configuration Elements"), which means that the order in which different entries appear inside the server configuration file is no longer random, but fixed by the XSD. As a result, the following element must be added immediately before the `<connection>` element in the server configuration file:

   ```
   <jobRepository>
   <property name="dbuser" value="dbuser"/>
   <property name="dbpassword" value="csf:reports:dbpasswdKey"/>
   <property name="dbconn" value="dbconn"/>
   </jobRepository>
   ```

   For information about adding a password key in the credential store, see Section 15.1.3, "Credential Store".

2. Save the server configuration file.

   Alternatively, you can configure the database job repository through Oracle Enterprise Manager:

   a. Navigate to the Reports Application home page in Enterprise Manager.

   b. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.

   c. Navigate to reports server mbean.

      Inprocess server - `oracle.reportsApp.config:Location=<managedServername>,name=rwserver,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3`

   d. In **Operations** tab click **addJobRepository**.

      It will add a child mbean called ReportsApp.JobRepository

   e. Click the mbean and use operation **addProperty** to add required properties - Username, Password Key and Database fields.

3. Configure a folder to which all instances have access (write access is needed, read access is not enough) by adding the `CacheDir` property to the `<cache>` element of each of the server configuration files,

   For example,

   on Windows:

   ```
   <property name="CacheDir" value="folder_name"/>
   ```

   on UNIX:

   ```
   <property name="CacheDir" value="/net/machine_name/usrs/tmp"/>
   ```

   where, `"/usrs/tmp"` is a shared location when Reports Server is running on different machines and has write privileges.

Note: ■CacheDir can also be used for high availability for sharing the cache.

- In a case where the in-process Reports Servers are running on different machines, the configuration can be done by sharing a folder to the other machine or user. For example, on Windows, configure the shared cache for both in-process Reports Servers by adding the following property to the `<cache>` element of each of the server configuration files:

  ```
  <property name="CacheDir" value="\\host\shared"/>
  ```

  where *host* is the machine where the shared folder is available, and `shared` is the folder name

For information about the `CacheDir` properties (new in Oracle Reports 12*c* Release (12.2.1.3), refer to Chapter 7, "Configuring Oracle Reports Services".

4. Load the `rw_server.sql` file to a database (this file is included with your Oracle Reports Services installation: `ORACLE_HOME\reports\admin\sql)`

   This creates a schema that owns the report queue information and has execute privileges on the server queue API.

5. Restart the Reports Server.

6. Set the Environment variables `TNS_ADMIN` and `ORACLE_HOME` in the shell and start the node manager (`$DOMAIN_HOME/bin/startNodeManager.sh`). This is required for the reports in-process server to connect to the database. See "TNS_ADMIN"

Note: The following are not supported in Reports High Availability:

- `rwclient` is not supported in Reports High Availability.
- *namingService* is not supported for Reports server discovery in High Availability setup. Only *multicast* is supported. For more information about High Availability about High Availability for Reports, see *High Availability Guide*.

3

Verifying Your Installation

After installing Oracle Reports, read through the section in this chapter to verify you are ready to use Oracle Reports Services to publish your reports:

- Understanding the Oracle Fusion Middleware Installation Structure
- Verifying OOTB Installation
- Verifying the Reports Server Environment
- Confirming Security with OPSS based Security
- Upgrading from the Prior Release

3.1 Understanding the Oracle Fusion Middleware Installation Structure

In prior releases, Oracle Reports Services was installed in a single Oracle home (`ORACLE_HOME`), including all files (both binaries and configuration). Oracle Fusion Middleware 12*c* introduced the option to split installation into an Oracle home (`ORACLE_HOME`) and an Domain Home (`DOMAIN_HOME`) to separate binaries from configuration, and allow shared binaries across servers.

The new Oracle Fusion Middleware installation option provides several advantages:

- `ORACLE_HOME` contains binary files, which can be shared across multiple `DOMAIN_HOMEs`, reducing the footprint and management of each `DOMAIN_HOME`.
- It is easier to apply patches and maintain the binary `ORACLE_HOME` without needing to reconfigure.
- Install of each `DOMAIN_HOME` is easier and lightweight.
- Easier backup, restore, and cloning.
- Support for Reports-only install.

For complete information about the new installation structure, including how to create a new Oracle instance, refer to *Administering Oracle Fusion Middleware*.

3.2 Verifying OOTB Installation

Make sure Oracle Reports components are created out-of-the-box (OOTB), and are available and ready to use after installation and configuration in the following way:

Start a standalone Reports Server, shut down the standalone Reports Server, and run a report using `rwservlet` and the in-process Reports Server.

User needs to create a reports server and reports tools using WLST command after OOTB installation.

3.3 Verifying the Reports Server Environment

Oracle Reports Services report requests flow from the Oracle HTTP Server component, to Oracle Reports Servlet, to Reports Server. Before sending report requests to Reports Server, verify that the environment is up and running:

- Checking Oracle HTTP Server
- Checking Oracle Reports Servlet
- Checking Reports Server

3.3.1 Checking Oracle HTTP Server

Before starting Reports Server, you must verify that your Oracle HTTP Server is running. For more information about performing this task in Oracle Enterprise Manager, refer to your Oracle Enterprise Manager documentation.

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

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

3.3.2 Checking Oracle Reports Servlet

To verify that Oracle Reports Servlet (`rwservlet`) is running, navigate to the following URL:

`http://host:port/reports/rwservlet/help`

where

`host` is the server that is allotted.

`port` is either the OHS port or the WebLogic Server port.

In a Single Sign-On environment, when using Oracle Access Manager 11g (OAM) as the authentication server, if `AUTHID` is passed in the Reports URL, then OAM authentication page is displayed.

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

3.3.3 Checking Reports Server

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

`http://host:port/reports/rwservlet/getserverinfo?server=server_name`

where

`host` and port are as described in Section 3.3.2, "Checking Oracle Reports Servlet".

`server=server_name` is not required if you are using the default Reports Server name (`rep_machine_name`) or the Reports Server specified in the Oracle Reports Servlet configuration file (`rwservlet.properties`).

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

Note: For more information about the Oracle Reports Servlet configuration file (`rwservlet.properties`), see Section 7.3, "Oracle Reports Servlet Configuration File".

3.4 Confirming Security with OPSS based Security

Confirm that security is enabled in the following way:

- Using the Command Line

3.4.1 Using the Command Line

On the command line, navigate to the following directory to open the in-process Reports Server configuration file to verify it is configured with the RWJAZN security:

DOMAIN_HOME`/config/fmwconfig/servers/WLS_REPORTS/applications/reports_`*version*`/configuration`

The presence of "`<security id="rwJaznSec" class="oracle.reports.server.RWJAZNSecurity"/>`" in the configuration file confirms that RWSecurity is enabled.

3.5 Upgrading from the Prior Release

- Upgrading from Oracle Reports 11.1.2 and 11.1.1 to 12*c* Release (12.2.1.3) is fully automated:
  - Integrated with Oracle Fusion Middleware Upgrade framework.
  - Automated configuration reduction mapping.
  - Extensive logging and diagnostics.
  - No loss of functionality.

Note: For information about installing Oracle Reports as a non default user see "Installing Oracle Forms and Reports as a Non-Default User" section in *Installation Guide for Oracle Forms and Reports*.

3.5.1 Backward Compatibility and Interoperability

Oracle Reports 12*c* Release (12.2.1.3) is fully backward compatible and interoperable with 11.1.2/11.1.1.

- 11.1.2/11.1.1 server/client compatible with 11g Release 2 (11.1.2) server/client.
- 11.1.2/11.1.1 reports can be run without any changes or loss of functionality.
- Support for Oracle Portal and Oracle Internet Directory based security along with new Java EE security.
- Support for 11.1.2. Toolkit-based font model along with new Java-based font model.
- Support for interoperability with 11.1.2. Oracle Forms Services, Oracle Portal, and Oracle Internet Directory.

4

Interoperability Scenarios and Considerations

This chapter discusses interoperability scenarios and considerations for Oracle Reports 12*c* Release (12.2.1.3).

It includes the following sections:

- Interoperability with Previous Versions of Oracle Reports
- Interoperability with Other Oracle Components

4.1 Interoperability with Previous Versions of Oracle Reports

Oracle Reports 12*c* Release (12.2.1.3) interoperates with Oracle Reports 11.1.2 and 11.1.1.

4.2 Interoperability with Other Oracle Components

Oracle Reports 12*c* Release (12.2.1.3) interoperates with other Oracle components, such as Oracle Portal, Oracle Internet Directory, Oracle Forms, Oracle BPEL Process Manager, and Oracle Application Server WebCache.

The following are the interoperability scenarios:

- Oracle Reports Server/Servlet with Oracle Portal
- Oracle Reports Server/Servlet with Oracle Internet Directory
- Oracle Reports Server with Oracle Forms Client
- Oracle Reports Server with Oracle Forms
- Oracle Reports Server with Oracle BPEL Process Manager
- Oracle Reports with Oracle WebCache

See:

- *Oracle Fusion Middleware Upgrade Guide for Oracle Portal, Forms, Reports, and Discoverer*
- *Oracle Fusion Middleware Installation Guide for Oracle Portal, Forms, Reports and Discoverer*

5

Starting and Stopping Oracle Reports Services

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

- Starting and Stopping Reports Server
- Starting, Stopping Reports Bridges from the Node Manager using script
- Starting Reports Components After Shutting Down an Instance
- Starting and Stopping the COS Naming Service
- Starting and Stopping the In-process Reports Server Using Oracle Reports Servlet
- 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 Oracle Fusion Middleware is installed. This includes Oracle Reports Services.

5.1 Starting and Stopping Reports Server

The best way to run Reports Server is through the Oracle WebLogic Scripting Tool (WLST). Node Manager provides a centralized mechanism for initializing, maintaining, and shutting down your Oracle HTTP Server, Oracle WebLogic Server processes, and OracleAS Reports Services. For more information on configuring Reports Server through Node Manager, see Section 7.8, "Configuring Reports Server with the Node Manager Reports Process Management"

Important: Node Manager automatically restarts Reports Server if it stops responding for some reason.

In Oracle Reports, running Reports Server as Windows service is no longer supported (`rwserver -install server_name`). As a result, the related command line keywords `INSTALL` and `UNINSTALL` are obsolete.

For more information about the obsolescence of running Reports Server as a Windows service, see *A Guide to Functional Changes Between Oracle Reports 6i and 10g* on the Oracle Technology Network (OTN).

5.1.1 Starting, Stopping Reports Servers from the Node Manager using script

To start and stop reports server, please use below UNIX commands:

```
$DOMAIN_HOME/bin/startComponent.sh reports_server_name

$DOMAIN_HOME/bin/stopComponent.sh reports_server_name
```

5.1.2 Alternative Methods of Starting and Stopping Reports Server

If you choose not to run Reports Server through Node Manager, you can use these older methods of running Reports Server:

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

> **Important:** Beginning with Oracle Reports 10g Release 2 (10.1.2), running Reports Server as a Windows service is no longer supported, as mentioned at the beginning of this section.

5.1.2.1 Starting the In-process Server (Windows and UNIX)

If you are using Reports Server as an in-process server (the default configuration), sending a run report request starts the in-process server; however, if you are sending a request through 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 server.

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

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

5.1.2.2 Starting Reports Server from a Command Line (Windows and Linux)

Before you start the Reports Server from command line, you must set the `COMPONENT_CONFIG_PATH` environment variable as follows:

```
COMPONENT_CONFIG_PATH=${DOMAIN_
HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_
name>
```

To start Reports Server as a standalone server on Windows, use 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
```

> **Note:** Before starting the Reports Server from the command line, ensure that the Reports Server is created if it does not already exist. For information about creating a Reports server, see Section 7.8.4, "Creating a New Reports Server Component Type".

Important: If DISPLAY is not set, you must start Reports Server in batch mode (batch=yes).

For more information about removing DISPLAY and printer dependencies in UNIX systems, see Section 10.8.

For more information about the REPORTS_DEFAULT_DISPLAY environment variable, see Appendix B.1.43.

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

5.1.2.3 Stopping Reports Server

There are several ways to stop Reports Server on Windows and UNIX, as follows:

- If Reports Server is running on Windows through the rwserver executable, or on UNIX through a shell script, rwserver.sh, click Shutdown in the Reports Server dialog box.
- If 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?authid=admin_user/admin_password
  ```

- If Reports Server is running from a command line on Windows or UNIX, use any of the following commands, depending on how you want to shut down the Reports Server.

  Note: On UNIX, use rwserver.sh instead of rwserver.

  Before you shut down the server, you must set the COMPONENT_CONFIG_PATH environment variable as follows:

  ```
  COMPONENT_CONFIG_PATH=${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>
  ```

  To shut down the server normally (that is, finish pending jobs and then stop):

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

  To shut down the server immediately (that is, stop without finishing pending jobs):

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

  To shut 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 Keywords"

Note: `authid` is Reports Server's administration user name and password. In Oracle Reports, the default security is based on standards-based Java EE security model through Oracle Platform Security Services. 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 must 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 and encrypted by Reports Server. You can reset the identifier element to any value. To change 'identifier' perform 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 encrypted="no">scott/tiger</identifier>
     ```

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

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

     Reports Server will now encrypt the username/password value of the identifier element. After Reports Server reads the changes made in the server_name.conf, 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
     ```

5.2 Starting, Stopping Reports Bridges from the Node Manager using script

The Oracle Reports bridge is used to connect two subnets. It acts as a gateway between Oracle Reports components running in different subnets.

For troubleshooting scenarios and diagnosis, see Appendix D.8, "Diagnosing Oracle Reports Bridge Problems".

5.2.1 Starting, Stopping, and Restarting the Oracle Reports Bridge from the Oracle Process Manager and Notification Server

To start and stop reports bridge, please use below UNIX commands:

```
$DOMAIN_HOME/bin/startComponent.sh reports_bridge_name
```

```
$DOMAIN_HOME/bin/stopComponent.sh reports_bridge_name
```

5.2.2 Starting and Stopping the Oracle Reports Bridge from the Command Line

It is recommended that you use WLST to start and stop Oracle Reports components.

You must set the following environment variable before you start or stop the Reports Bridge component:

```
COMPONENT_CONFIG_PATH=${DOMAIN_
HOME}/config/fmwconfig/components/ReportsBridgeComponent/<reports_bridge_
name>
```

To start the Oracle Reports bridge from the command line, use the following commands:

On Windows:

```
rwbridge.bat name=bridgename
```

On UNIX:

```
rwbridge.sh name=bridgename
```

For example, to start an Oracle Reports bridge named `foo` on Windows, use the following command:

```
rwbridge.bat name=foo
```

For more information about the rwbridge executable, see Section A.2.7, "rwbridge"

Oracle Reports creates a configuration file, *`rwbridge.conf`* when the Oracle Reports bridge is started for the first time. This file is generated based on the settings in the `rwbridge.template` file and is located in the `$DOMAIN_HOME/config/fmwconfig/components/ReportsBridgeComponent/<bridge_name>/rwbridge.conf` directory. Edit the *`rwbridge.conf`* file to specify remote Oracle Reports bridges to connect to other subnets.

Note: You must restart the Oracle Reports bridge for any configuration changes to take effect.

To stop an Oracle Reports bridge, use the following command:

On Windows:

```
rwbridge.bat name=bridgename shutdown=normal authid=username/password
```

On UNIX:

```
rwbridge.sh name=bridgename shutdown=normal authid=username/password
```

For example, to stop an Oracle Reports bridge named `foo` on UNIX, use the following command:

```
rwbridge.sh name=foo shutdown= normal authid=scott/tiger
```

In the configuration file, repbrg_bridgename.conf, modify the identifier element to specify the *`username/password`* and set the `encrypted` attribute to no. This is to indicate that the password is not encrypted. This password will be encrypted once the Oracle Reports bridge is started.

For example:

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

Usage Notes

- If the identifier element is commented, then it is possible to stop the Oracle Reports bridge without specifying `authid`

- It is not possible to stop the Oracle Reports bridge remotely.

See Section 7.2.1.18, "identifier"

5.3 Starting Reports Components After Shutting Down an Instance

When a Reports instance is shut down, you can bring up the Reports components as follows.

5.3.1 Starting Reports Servlet

To start Reports Servlet in an expand cluster, complete the following steps:

1. Start the Node Manager.
   a. From your present working directory, run the following command:

      ```
      cd $DOMAIN_HOME/bin
      ```

   b. Run the following command:

      ```
      ./startNodeManager.sh
      ```

2. Start the Admin Server.
   a. From your present working directory, run the following command:

      ```
      cd $DOMAIN_HOME/bin/
      ```

   b. Run the following command to start the admin server:

      ```
      ./startWebLogic.sh
      ```

3. Start `WLS_REPORTS` from the WebLogic Server Administration Console.
   a. Log in to the WebLogic Server Administration Console. (`http://host:7001/console`).
   b. From the **Domain Structure** section in the left navigation pane, select **Environment** > **Servers**.The **Summary of Servers** screen is displayed. Click the **Control** tab.
   c. Select the **WLS_REPORTS** check box and click **Start**.
   d. From the **Domain Structure** section in the left navigation pane, select **Deployments**.

      The **Summary of Deployments** screen is displayed.
   e. Click **Next** to navigate to the screen that lists the Reports Application.
   f. Click **Start**.

5.3.2 Starting Reports Standalone Server

To start and stop reports server, please use below commands:

```
$DOMAIN_HOME/bin/startComponent.sh reports_server_name
```

```
$DOMAIN_HOME/bin/stopComponent.sh reports_server_name
```

5.4 Starting and Stopping the COS Naming Service

The Common Object Service (COS) naming service orbd, provided by Sun Microsystem's JDK, can be used for Reports Server discovery instead of the default

broadcast mechanism. Refer to the JavaIDL page on Sun Microsystem's Web site (http://www.oracle.com/technetwork/java/index.html) for more details on the orbd executable.

To start the naming service, use the following commands:

On Windows:

```
namingservice.bat port_number
For example: DOMAIN_HOME\reports\bin\namingservice.bat
```

On UNIX:

```
namingservice.sh port_number
For example: DOMAIN_HOME/reports/bin/namingservice.sh
```

5.5 Starting and Stopping the In-process Reports Server Using Oracle Reports Servlet

If you are using Reports Server as an in-process Reports Server (the default configuration), sending a run report request starts the in-process Reports Server. However, if you are sending a request through a command line, Oracle Reports Servlet (rwservlet) must be started first using either the run report URL or the Web command URL. When you have successfully started rwservlet, you have successfully started the in-process Reports Server.

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

```
http://machine_name:port/reports/rwservlet/startserver
http://machine_name:port/reports/rwservlet/stopserver
```

5.6 Verifying that the Oracle HTTP Server Is Running

OracleAS Reports Services depends upon the Oracle HTTP Server component. Before starting Reports Server through WLST, you must verify that your Oracle HTTP Server is running. For more information about performing this task in Oracle Enterprise Manager, 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/
```

5.7 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 Reports Server is running, navigate to the following URL:

```
http://your_machine_name.domain_name:your_port_
number/reports/rwservlet/getserverinfo?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 ($DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_

`<version>/configuration`). If this URL executes successfully, you should see a the Reports Server information.

Note: You'll find more information about the servlet configuration file in Section 7.3, "Oracle Reports Servlet Configuration File".

Part II

Administering Oracle Reports Services

Part II provides information about administering Oracle Reports Services:

6

Administering Oracle Reports Services Using Oracle Enterprise Manager

Reports EM Administration is predominantly done using EM Mbean browser. Oracle Reports 12*c* Release (12.2.1.3) is integrated with Oracle WebLogic Server, which results in simpler administration of complex topology and deployments. You can manage and monitor Oracle Reports components using either:

- Oracle Enterprise Manager, included with Oracle Fusion Middleware.
- Oracle WebLogic Scripting Tool (WLST)

This chapter describes how to manage and monitor Oracle Reports components using Oracle Enterprise Manager. It includes the following main sections:

> **See Also:** For more information on Oracle Enterprise Manager, refer to the *Administering Oracle Fusion Middleware*, available on the Oracle Fusion Middleware documentation CD.

6.1 Configuring Oracle Reports Components

Reports configuration files and their mbean locations are given below. The mbeans can be accessed from EM mbean browser

Weblogic Domain > System MBean Browser

Table 6–1 Configuration Location

Component/ Integration area	Location, File Name	Central/Local
Reports In-process server / Servlet config files	${DOMAIN_HOME}/config/fmwconfig/servers/<managedServerName>/applications/reports_12.2.1.3/configuration/* cgicmd.dat, logging.xml, rwserver.conf, xmlpds.conf, logmetadata.xml, rwservlet.properties, jdbcpds.conf rwnetwork.conf, textpds.conf Mbean - oracle.reportsApp.config:Location=<managedServername>,name=<configFile>,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3	Local
	ODL logging config ${DOMAIN_HOME}/config/fmwconfig/servers/<managedServerName>/logging.xml Mbean - oracle.logging:type=LogConfig,ServerName=<managedServerName>	Central
Reports Server config files	${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/* rwserver.conf, xmlpds.conf, jdbcpds.conf, rwnetwork.conf, textpds.conf Mbean - oracle.reports.serverconfig:type=ReportsServer,name=<configFile>-<componentName>	Central
	ODL logging config ${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/logging.xml Mbean - oracle.logging:type=LogConfig,componentType=oracle_repserv,name=<componentName>	Central
	${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/ components-logs.xml	Internal file Central

Table 6–1 (Cont.) Configuration Location

Component/ Integration area	Location, File Name	Central/Local
Reports Tools config files	${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/* rwnetwork.conf, xmlpds.conf, jdbcpds.conf, rwbuilder.conf, textpds.conf Mbean - oracle.reports.toolsconfig:type=ReportsTools,name=<configFile>-<componentName>	Central
	ODL logging config ${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/logging.xml Mbean - oracle.logging:type=LogConfig, componentType=oracle_reptools,name=<componentName>	Central
	${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/ components-logs.xml	Internal file Central
Reports Bridge config files	${DOMAIN_HOME}/config/fmwconfig/components/ReportsBridgeComponent/<reports_bridge_name>/* rwbridge.conf, rwnetwork.conf Mbean - oracle.reports.bridgeconfig:type=ReportsBridge,name=<configFile>-<componentName>	Central
	ODL logging config ${DOMAIN_HOME}/config/fmwconfig/components/ReportsBridgeComponent/<reports_bridge_name>/logging.xml Mbean - oracle.logging:type=LogConfig, componentType=oracle_repbrd,name=<componentName>	Central
	${DOMAIN_HOME}/config/fmwconfig/components/ReportsBridgeComponent/ components-logs.xml	Internal file Central

Table 6–1 (Cont.) Configuration Location

Component/ Integration area	Location, File Name	Central/Local
Reports Shell scripts/ Bat files	${DOMAIN_ HOME}/reports/bin/*.sh (Unix) namingservice.sh, rwbuilder.sh, rwdiag.sh, rwserver.sh, reports.sh, rwclient.sh, rwengine.sh, rwrqv.sh, rwbridge.sh, rwconverter.sh, rwlpr.sh, rwrun.sh ${DOMAIN_ HOME}/reports/bin/*.bat (Windows) namingservice.bat, rwbuilder.bat, rwdiag.bat, rwserver.bat, reports.bat, rwclient.bat, rwrqm.bat, rwbridge.bat, rwconverter.bat, rwrun.bat	Local
Reports Printer / Font config (Formerly called Forms Reports common configuration)	${ReportsToolsConfig} refers to location ${DOMAIN_ HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name> Font Setup: Font aliasing / Pdf - subsetting/embedding/aliasing font configuration ${ReportsToolsConfig}/guicommon/tk/admin/Uifont.ali Mbean - oracle.frcommon.config:type=uifont.ali,name=uifont-<componentName>	Central
	Printer Configuration (Unix only) ${ReportsToolsConfig}/guicommon/tk/admin/uiprint.txt ${ReportsToolsConfig}/guicommon/tk/admin/uiscreenprint.txt Mbean - oracle.frcommon.config:type=uiprint.txt,name=uiprint-<componentName> Mbean - oracle.frcommon.config:type=uiscreenprint.txt,name=uiscreenprint-<componentName>	Central

Table 6–1 (Cont.) Configuration Location

Component/ Integration area	Location, File Name	Central/Local
	Toolkit config, Encoding etc (Unix only) ${ReportsToolsConfig}/guicommon/tk/admin/Tk2Motif.rgb Mbean - oracle.frcommon.config:type=Tk2Motif.rgb,name=Tk2Motif-<componentName>	Central
	PPD files (Unix only) ${ReportsToolsConfig}/guicommon/tk/admin/PPD/* Mbean - oracle.frcommon.config:type=screenprinter.ppd,name=screenprinter-<componentName>	Central
	AFM (Unix only) ${ReportsToolsConfig}/guicommon/tk/admin/AFM/*	Central
Reports Process Management Config (Reports Server / Reports Bridge	{componentType} - is ReportsServerComponent / ReportsBridgeComponent for Reports Server / Reports Bridge respectively {componentName} - is name of component Managed by nodemanager-startup properties ${DOMAIN_HOME}/system_components/{componentType}/{componentName}/data/nodemanager/startup.properties	Local
	Reports environment and timeout config ${DOMAIN_HOME}/system_components/{componentType}/{componentName}/data/nodemanager/reports_process.xml	Local
Reports OHS Integration Config file	${DOMAIN_HOME}/config/fmwconfig/components/OHS/instances/<ohsName>/moduleconf/reports_ohs.conf	Central
OPSS config files (FMW Level)	${DOMAIN_HOME}/config/fmwconfig/jps-config.xml - J2EE OPSS config file ${DOMAIN_HOME}/config/fmwconfig/jps-config-jse.xml: J2SE OPSS config file ${DOMAIN_HOME}/config/fmwconfig/system-jazn-data.xml - XML ID/Policystore	Central

6.1.1 Configuring a Mail Server

Oracle Reports 12*c* Release (12.2.1.3) provides the option to configure the Mail Server element from Enterprise Manager. To configure the Mail Server element, complete the following steps:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser
3. Navigate to reports server mbean

   ```
   Standalone server -
   oracle.reports.serverconfig:type=ReportsServer,name=rwserver-<component
   Name> Inprocess server -
   oracle.reportsApp.config:Location=<managedServername>,name=rwserver,typ
   e=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3
   ```

4. Go to **Operations** and click **addPluginParam**.
5. Give Name as **mailServer**.
6. Click **Invoke**.
7. Click the child mbean which is created.
8. Click **Operations** > **Add Property** and add properties as needed and click **Invoke** for each property
   - enableSSL
   - mailUserName
   - mailPassword

   For details of these properties, please refer to Section 7.2.1.2, "pluginParam"

6.2 Administering and Scheduling Jobs

Oracle Enterprise Manager does not have the facility to view or manage reports jobs. Instead please use reports servlet commands for the same

6.2.1 Displaying Jobs

To retrieve information about one or more jobs (for example, status, errors for failed jobs, trace information):

1. Use reports servlet showjobs/showmyjobs commands.

   Refer to Section A.8.8, "SHOWJOBS" and Section A.8.10, "SHOWMYJOBS" for more details.

6.2.2 Displaying a Consolidated Job Queue

For displaying consolidated job queue in high availability (HA) configuration you can use `rwservlet` and the `SHOWJOBS` Web command to display a consolidated job queue, as shown in Section A.8.8, "SHOWJOBS".

Note:

- By default, the jobs of all the nodes that are participating in the high availability (HA) configuration are shown. This makes it easy to administer the consolidated job queue (for example, for killing a particular job, or getting the information about a job). The administrator does not have to view the job queues for each Reports Server separately.
- It is possible to see the jobs of a particular Reports Server by choosing the server name in the drop-down list. It does not matter which node the load balancer directs the request to, you can always see the job information of any Reports Server participating in the group, even if one or more of the Reports Servers in the group are down. This is made possible by the database-based shared job repository. Additionally, each of the Reports Servlets (corresponding to each in-process Reports Server) contains the list of the other in-process Reports Servers that are part of the HA scenario.

6.2.3 Performing Operations on Jobs

To perform operations on jobs, such as resubmitting (retrying) a job you can use reports servlets showjobs page.

6.2.4 Scheduling Jobs

To define job scheduling options user reports servlet to schedule jobs. Refer to Section A.8.3, "SCHEDULE" for more details.

6.3 Securing Oracle Reports Services

In 12*c* Release (12.2.1.3), Reports Server is secure out-of-the-box using the Oracle Platform Security Services, which accomplishes both authentication and authorization. Oracle Reports uses this Java EE-based security model to allow you to create security policies for running report jobs and Web commands.

In prior releases, Reports Server authentication was restricted to use only Oracle Internet Directory. Authorization of Reports Server required an Oracle Portal-based security model (using Portal metadata repository for checking authorization). If you want to revert to the security mechanism of prior releases, refer to Section 6.3.1.1, "Switching to Oracle Portal Security".

In Oracle Reports 12*c* Release (12.2.1.3), administrators can use Oracle Enterprise Manager to more easily define and manage granular security policies and file system access:

- Enabling and Disabling Security and Changing Security Mechanism used
- Defining Security Policies for Reports
- Defining Security Policies for Directories
- Defining Security Policies for Web Commands
- Defining Read/Write Access to Directories
- Enabling and Disabling Single Sign-On

- Using Oracle Access Manager
- Managing Credentials

6.3.1 Enabling and Disabling Security and Changing Security Mechanism used

To enable or disable security for the Reports Server or Reports Application:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser
3. Navigate to reports server mbean

   ```
   Standalone server - Folder: Configuration Beans -->
   Folder:oracle.reports.serverconfig --> type=ReportsServer,
   name=rwserver-<componentName>, ReportsServer.Job
   ```

   ```
   Inprocess server - Folder: Application Defined MBeans -->
   Folder:oracle.reportsApp.config -->
   Server:<managedServername>,Application=reports, type=ReportsApp,
   name=rwserver, ReportsApp.Job
   ```

4. Click child mbean: rwEngrwJaznSec
5. For property securityId enter value as follows:
   - `rwJaznSec` - for OPSS based security
   - `rwSec` - for Portal based security
   - `<Leave Empty>` - Unsecure
6. Click *Apply*

6.3.1.1 Switching to Oracle Portal Security

The steps for deploying reports in Oracle Portal are the same in this release as in prior releases, as described in Chapter 16, "Deploying Reports in Oracle Portal". However, the security mechanism underlying the deployment has changed. In that, authorization is enabled out of the box, but during installation if only Oracle Internet Directory is specified and Portal is not installed, authorization using Oracle Portal is disabled. The default installation accomplishes both authentication and authorization through Oracle Platform Security Services

You can continue to use the security features in Oracle Portal from prior releases for backward compatibility. To switch from the new Oracle Platform Security Services to pre-11*g* Oracle Portal metadata repository-based security:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to reports server mbean

   Standalone server -
   `oracle.reports.serverconfig:type=ReportsServer,name=rwserver-<componentName>`

   Inprocess server -
   `oracle.reportsApp.config:Location=<managedServername>,name=rwserver,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3`

4. Click child `mbean ReportsServerJob`
5. For property securityId enter value as follows:

`rwSec` - for Portal based security

Note: If you enable Oracle Portal security features, then Oracle Portal must also be configured during installation for authorization to occur:

- If Oracle Portal is configured during installation, authentication is accomplished using the Oracle Internet Directory and authorization is accomplished using Oracle Portal (which stores authorization policies).
- If Oracle Portal is *not* configured during installation, authentication is accomplished using the Oracle Internet Directory and authorization does not occur.

6.3.2 Defining Security Policies for Reports

As administrator, you can specify the reports to which a particular user/role has access by creating security policies for each report. In the security policy, you can also specify the server, destination name (`desname`), destination type (`destype`), and other parameters. An authenticated user is authorized against these security policies.

To define security policies for reports for Reports Server or Reports Application (in-process Reports Server):

1. Log in to Oracle Enterprise Manager.
2. Navigate to **Weblogic Domain** > **Security** > **Application Policies**.
3. Select Application Stripe as `reports`
4. Click **Search** icon.
5. Enter appropriate values for the elements on the page to define or edit a security policy using the descriptions in the Help topic for the page.
6. Click online help for the page to access the page-level help.
7. Click **Create** to navigate to **Create Application Grant** page.
8. In the **Grantee** section add the reports roles to assign permissions to.
9. In the **Permissions** section add the new permission granted to the roles.

   The permission class to be used is "oracle.reports.server.ReportsPermission"
10. As **Resource Name** give the command line to give permission to. User can also use wildcard ("*") as parameter values.
11. As **Permission Actions** give "*" as the value

Note: The security policies defined in Oracle Enterprise Manager are stored in the policy store configured by the user. The idstore contains information on the users and the policy store contains the security policies configured by the user.

6.3.3 Defining Security Policies for Directories

In certain cases, you will want to give a particular user access to multiple related reports. Rather than specify a security policy for each report, you can collect all the

reports in a single directory, then specify a security policy for the directory. Again, the security policy is checked when the user provides the user name and password.

As an example, imagine that there are 15 finance reports, for which you want to give access to the FINANCE role, and there are 12 Human Resources reports for which you want to give access to the HR role. Rather than specify 15 security policies for FINANCE role, and 12 policies for HR role (one policy per report), you can collect all finance reports in one directory, and collect all the HR reports in another directory, then specify only 2 policies (one per directory). Instead of specifying the report name, you will specify the directory name in the security policy.

To define a security policy for directories:

1. Use the steps in Section 6.3.5, "Defining Read/Write Access to Directories" to define security policies. There you can also define security policies for directories.
2. Run a report as the specified role and other roles to test that security policies for authentication and authorization are enforced as you have defined. For example, run a report from your browser using the following URLs:

   ```
   http://host:port/reports/rwservlet?report=report_
   name.rdf&destype=cache&desformat=html&userid=user/password@mydb&server=ReportsS
   erver_instancename

   http://host:port/reports/rwservlet?report=report_
   name.rdf&userid=user/password@mydb&destype=file&desformat=pdf&desname=report_
   name.pdf
   ```

   where

   host is the machine where the Oracle Instance is set up

   port is the OHS main port
3. In command line option 'report' in security policy enter one or more report definition file names or the directories for which you are defining security policies. For example, to specify a directory, enter: `/myreports/runtime/reports/*`. Separate multiple entries with a comma (,).

Now, to use the defined directory access control at the Reports Server level, refer to Section 6.3.1, "Enabling and Disabling Security and Changing Security Mechanism used" to confirm that security is turned on.

> **Note:** The security policies defined in Oracle Enterprise Manager are stored in the policy store configured by the user. The idstore contains information on the users and the policy store contains the security policies configured by the user.

6.3.4 Defining Security Policies for Web Commands

You can also specify the Web commands to which a particular user/role has access by creating security policies for each Oracle Reports Servlet (`rwservlet`) Web command. The security policy is checked when the user provides the user name and password.

To define security policies for Web commands:

1. Log in to Oracle Enterprise Manager.
2. Navigate to **Weblogic Domain** > **Security** > **Application Policies**.
3. Select Application Stripe as `reports`

4. Click **Search** icon.
5. Enter appropriate values for the elements on the page to define or edit a security policy using the descriptions in the Help topic for the page.
6. Click online help for the page to access the page-level help.
7. Click **Create** to navigate to **Create Application Grant** page.
8. In the **Grantee** section add the reports roles to assign permissions to.
9. In the **Permissions** section add the new permission granted to the roles.

   The permission class to be used is "oracle.reports.server.WebCommandPermission"
10. As **Resource Name** give the command line to give permission to. User can also use wildcard ("*") as parameter values. It will be of the form given below:

    webcommands=<webcommands list separated by comma> server=<reports server names>

    ```
    webcommands=showmyjobs,getjobid,showjobid,getserverinfo,showjobs server=*
    ```
11. As **Permission Actions** give `execute` as the value

> **Note:** The security policies defined in Oracle Enterprise Manager are stored in the policy store configured by the user. The idstore contains information on the users and the policy store contains the security policies configured by the user.

6.3.5 Defining Read/Write Access to Directories

As an administrator, you can specify read/write access for Reports Server, Reports Application (in-process Reports Server), or Oracle Reports Runtime to directories. This feature only checks whether Reports Server, Reports Application, or Oracle Reports Runtime is authorized to read from or write to a specified directory, and is unrelated to security policies that check the user name and password.

- **Read access**. To avoid the security issue of exposing sensitive content of files, you can specify the directories from which Reports Server, Reports Application, or Oracle Reports Runtime is allowed to read.

  For example, a malicious user may specify the following keywords to run a report on Windows:

  ```
  distribute=yes&destination=C:\Temp
  ```

  This would generate an error stating that there was an error in the syntax of the file. To avoid this, enable file system access control to specify read directories that do not include system directories.
- **Write access**. To avoid the security issue of a malicious user potentially overwriting a system file by sending report output to a system directory, you can specify the directories to which Reports Server, Reports Application, or Oracle Reports Runtime is allowed to write. Attempts to write to other directories will return an error.

  For example, a user may run a report to the following destination on Windows:

  ```
  desname=C:\Temp
  ```

This would overwrite a system file unless file system access control was enabled to specify write directories that do not include system directories.

To define read/write access to directories for Reports Server, Reports Application, or Oracle Reports Runtime:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to reports server mbean:

   Standalone server - `oracle.reports.serverconfig:type=ReportsServer,name=rwserver-<componentName>`

   Inprocess server - `oracle.reportsApp.config:Location=<managedServername>,name=rwserver,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3`
4. Click **Operations**.
5. Click **addFolderAccess**
6. Enter the names of the Read Directories and Write Directories to which Reports Server, Reports Application, or Oracle Reports Runtime should have access. These entries set the read and write sub-elements of the folderaccess element in the configuration file.

   Read Directories: To avoid the security issue of exposing sensitive content of files, enter the names of the directories from which Reports Server is allowed to read. Separate directory names with a semicolon (;).

   Write Directories: Enter the names of the directories to which Reports Server is allowed to write. Attempts to write to other folders will return an error.
7. Click **Invoke**.

6.3.5.1 Errors when Running Reports using DESTYPE=FILE in Oracle Reports

This section explains how to resolve errors when trying to run Reports with DESTYPE=FILE in Oracle Reports.

The errors you may encounter:

- `REP-69: An internal error occurred`
- `REP-56133: Access is denied to write to the specified location`

The rwserver_diagnostic.log file shows the following message:

```
[2015-09-30T21:54:02.715+01:00] [WLS_REPORTS] [ERROR] [] [oracle.reports.server]
[tid: 152] [userId: <anonymous>] [ecid: 0000L0W27NFCgo2_vpT4iX1M34jQ000002,0:1:3]
[APP: reports] [partition-name: DOMAIN] [tenant-name: GLOBAL] SecurityHelper:start
Writeable folders not configured. Output destination to 'File' will fail.
```

In Oracle Reports, the Reports Servers are writable folder secured out-of-the-box. This means that in order to use DESTYPE=FILE parameter, the reports server needs to have defined which folders can be used to write reports output to before using them. If you run Reports using DESTYPE=FILE in Oracle Reports 11g, you will not encounter this problem.

To resolve these errors, perform the following steps to enable **folderAccess** setting in Reports Server configuration file:

Note: If **auto-commit Mode** is disabled, in the Oracle Enterprise Manager console window go to the Change Center pane (Lock icon) and click **Lock & Edit** in order to be able to modify values via Mbean. When complete, click **Activate Changes** in the Change Center pane.

1. Log in to Fusion Middleware Control.
2. From the **WebLogic Domain** menu, select **System MBean Browser**.
3. Navigate to the MBean you want to configure:

   For Inprocess Server:

   - Expand **Application Defined MBeans**.
   - Click **oracle.reportsApp.config** > **Server: WLS_REPORTS** > **Application: reports** > **ReportsApp**.

   For Standalone Server:

   - Expand **Configuration MBeans**.
   - Click **oracle.reports.serverconfig** > **ReportsServer** > **rwserver-[standalone name]**.
4. Click **rwserver** and then select **Operations** tab.
5. Click **addFolderAccess** operation.
6. In the **Write** row, enter the name of the folder(s) in the **Value** field, to which Reports Server is allowed to write.

   For example:

   Windows: `c:\output1;d:\output2`

   Unix: `/u01/report;/u02/rdf`
7. Click **Invoke**.

   For example: In the *rwserver.conf file*, **folderAccess** is set as follows:

   Windows:

```
<folderAccess>
      <write>c:\output1;d:\output2</write>
</folderAccess>
```

   Unix

```
<folderAccess>
      <write>/u01/report;/u02/rdf</write>
</folderAccess>
```
8. Restart the Reports Server.

6.3.6 Enabling and Disabling Single Sign-On

If you plan to take advantage of Oracle Application Server Single Sign-On, you can use Oracle Enterprise Manager to set the `SINGLESIGNON` parameter in the `rwservlet.properties` configuration file. `SINGLESIGNON=YES` by default on installation. For more information about Single Sign-On, refer to Chapter 17, "Configuring and Administering Oracle Single Sign-On".

To enable Single Sign-On:

1. Log in to Oracle Enterprise Manager.

2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to reports servlet mbean

   ```
   oracle.reportsApp.config:Location=<managedServername>,name=rwservlet,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3
   ```

4. Set property **Singlesignon** as yes/no to enable/disable single signon.
5. Click **Apply**.

6.3.7 Using Oracle Access Manager

Oracle Access Manager is a component of Oracle Fusion Middleware that you can use in place of OracleAS Single Sign-On 10g to implement centralized authentication, policy-based authorizations, delegated administration, and so on.

You can use the Oracle Fusion Middleware Upgrade Assistant to upgrade from OracleAS Single Sign-On 10g to Oracle Access Manager 11*g*. For more information about upgrading to Oracle Access Manager 11*g*, see the "Upgrading Your Oracle Single Sign-On Environment" chapter in the *Oracle Fusion Middleware Upgrade Guide for Oracle Identity Management*.

6.3.8 Managing Credentials

This section explains how to use the oracle Enterprise Manager to manage credentials in a domain credential store.

1. Log in to Oracle Enterprise Manager and navigate to **WebLogic Domain** > **Security** > **Credentials**, to display the Credentials page.
2. Use the button **Delete** to remove a selected item (key or map) in the table. Note that deleting a credential map, deletes all keys in it. Similarly, use the button Edit to view or modify the data in a selected item.
3. To display credentials matching a given key name, enter the string to match in the box **Credential Key Name**, and then click the blue button to the right of it. The result of the query is displayed in the table.
4. To redisplay the list of credentials after examining the results of a query, select **WebLogic Domain** > **Security** > **Credentials.**

To add a new key to a credential map:

1. Click **Create Map** to display the **Create Map** dialog.
2. In this dialog, enter the name of the map for the credential being created.
3. Click **OK** to return to the **Credentials** page. The new credential map name is displayed with a folder icon in the table.

> **Note:** In CSF, the Reports Server can access credentials only from the **Reports** folder, hence you must create credentials under the **Reports** folder.

To add a new key to a credential map:

1. Click **Create Key** to display the **Create Key** dialog.
2. In this dialog, select a map from the pull-down list **Select Map** where the new key will be inserted, enter a key in the text box **Key**, select a type from the pull-down

list **Type** (the appearance of the dialog changes according to the type selected), enter the required data.

3. Click **OK** when finished to return to the Credentials page. The new key is shown under the map icon corresponding to the map you selected.

For more information about Reassociating the Credential Store, see *Securing Applications with Oracle Platform Security Services*.

6.4 Managing Fonts

In Oracle Reports 12*c* Release (12.2.1.3), administrators can use Oracle Enterprise Manager to simplify configuring fonts and diagnosing font issues:

- Configuring Fonts
- Diagnosing Font Issues

For more information about fonts, refer to Chapter 9, "Managing Fonts in Oracle Reports".

6.4.1 Configuring Fonts

To configure font subsetting and aliasing information:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to reports tools mbean for uifont.ali `oracle.frcommon.config:type=uifont.ali,name=uifont-<componentName>.`
4. Use the Printer Configuration and Font Configuration child mbeans to modify the `uifont.ali`.
5. Navigate to reports tools mbean for `screenprinter.ppd`

   `oracle.frcommon.config:type=screenprinter.ppd,name=screenprinter-<componentName>`
6. Use the Screenprinter Definition Parameters (Default Resolution and Default Font attributes) to modify the `screenprinter.ppd` file.

6.4.2 Diagnosing Font Issues

To diagnose issues with fonts in your report output:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to mbean

   Standalone server - `oracle.logging:type=LogConfig,componentType=oracle_repserv,name=<componentName>`

   Inprocess server - `oracle.logging:type=LogConfig,ServerName=<managedServerName>`
4. Click Operation `setLoggerLevel`

   For Logger Name - enter `oracle.reports.engine`

   For Logger Level - enter `TRACE:1`

5. Click **Invoke**.
6. Run your report again.
7. View log messages using WLST commands refer to Audit Configuration WLST Commands.

6.5 Monitoring Performance

Oracle Enterprise Manager does not have the facility to view or manage reports jobs. Instead please use reports servlet commands for the same.

Use reports `servlet showjobs/showmyjobs` commands.

Refer to Section A.8.8, "SHOWJOBS" and Section A.8.10, "SHOWMYJOBS" for more details.

Reports Server performance can be via `getserverinfo` command

Refer to Section A.6.11, "GETSERVERINFO" for more details.

6.6 Managing Log Files

Oracle Reports 12*c* Release (12.2.1.3) provides improved diagnosability through logging and tracing enhancements.

All Oracle Reports log files follow Oracle Diagnostic Logging (ODL) format, the standard across Oracle Fusion Middleware, for log format, message types, and log management directives. The log file entries are in Text format (default) or XML format. For detailed information, refer to *Administering Oracle Fusion Middleware*.

For information about log file enhancements, see Section 24.2.1, "Log Files"

Note: If you change the log path for the in-process server engine (that is, oracle.reports.engine logger), ensure that you make similar changes in the logmetadata.xml file. This file resides in the same directory as logging.xml.

6.6.1 Viewing and Searching Log Files

Viewing and searching log files can be done using WLST commands as explained in `https://docs.oracle.com/middleware/1213/core/ASADM/logs.htm#ASADM218`

To modify the information logged in log files to diagnose issues, see Section 6.13.1, "Specifying Logging Information".

6.6.2 Configuring Log Levels

To configure Log Levels in Oracle Enterprise Manager:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to mbean

   Standalone server - `oracle.logging:type=LogConfig,componentType=oracle_repserv,name=<componentName>`

   Inprocess server - `oracle.logging:type=LogConfig,ServerName=<managedServerName>`

Reports Tools - `oracle.logging:type=LogConfig,componentType=oracle_reptools,name=<componentName>`

Reports Bridge - `oracle.logging:type=LogConfig,componentType=oracle_repbrd,name=<componentName>`

4. Click Operation setLoggerLevel
5. For "Logger Name" enter logger name, one of the following:
   - oracle.reports.server
   - oracle.reports.servlet
   - oracle.reports.engine
6. For "Logger Level" enter log level
7. Click **Invoke**

6.7 Modifying Reports Server Audit Configuration

To modify Reports Server Audit Configuration in Oracle Enterprise Manager:

1. Log in to Oracle Enterprise Manager.
2. Navigate to Domain Home page
3. From the **WebLogic Domain** menu, click **Security** > **Audit Policy**.
4. Select **Component Name** as ReportsServer
5. From the **Audit Level** drop-down list, select **Custom.**
6. Select ReportsServer
7. Click the **User Sessions** and **Authorization** rows to edit the corresponding configuration settings.

6.8 Registering Pluggable Destinations with Reports Server

To register a pluggable destination with Reports Server:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to reports server mbean

   Standalone server - `oracle.reports.serverconfig:type=ReportsServer,name=rwserver-<componentName>`

   Inprocess server - `oracle.reportsApp.config:Location=<managedServername>,name=rwserver,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3`
4. In **Operations** click `addDestination`.
5. Enter appropriate **Name** and **Class** values for the new pluggable destination.

   For example, to register the `SecurePDF` pluggable destination:

   Name: `SecurePDF`

   Class: `oracle.reports.plugin.destination.securepdf.SecurePdfDestination`

6. You can also click **Add** to add the properties for the destination in the Registered properties.
7. Click **Invoke**.

   New child mbean with the destination name is created.

   Using those mbeans you can also use operation `addProperty` to add the properties for the destination

6.9 Configuring Proxy Information

To specify proxy information:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to reports server mbean

   Standalone server -
   `oracle.reports.serverconfig:type=ReportsServer,name=rwserver-<componentName>`

   Inprocess server -
   `oracle.reportsApp.config:Location=<managedServername>,name=rwserver,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3`
4. In child mbean ProxyInfo you can set proxy hosts and ports available for various protocols, such as http, https, ftp, or file. You can also specify the addresses for which proxies can be bypassed.

6.10 Managing and Monitoring a Reports High Availability (HA) Solution

Oracle Fusion Middleware consists of many components that can be deployed in distributed topologies. The underlying paradigm used to enable high availability for Oracle Fusion Middleware is clustering, which unites various Oracle Fusion Middleware components in certain permutations to offer scalable and unified functionality, and redundancy should any of the individual components fail. See Section 2.4, "Setting Up a High Availability Environment".

6.10.1 Configuring Reports Server for High Availability

Refer to Section 2.4.3, "Configuring Reports Server for High Availability".

6.10.2 Displaying a Consolidated Job Queue

Refer to Section 6.2.2, "Displaying a Consolidated Job Queue".

6.10.3 Specifying a Shared Cache Directory

To specify a shared cache directory for high availability (HA):

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to reports server mbean

Standalone server -
`oracle.reports.serverconfig:type=ReportsServer,name=rwserver-<componentName>`

Inprocess server -
`oracle.reportsApp.config:Location=<managedServername>,name=rwserver,type=ReportsApp,Application=reports,ApplicationVersion=12.2.1.3`

4. Click child mbean Cache.
5. In **Operations** tab use `addCacheDirProperty` method to enter the path to the shared cache directory for HA and click **Invoke**.

6.11 About the Oracle Fusion Middleware System MBean Browser

The Oracle Fusion Middleware System MBean Browser is a part of Oracle Fusion Middleware Control, and it is used to update configuration settings for middle tier components.

This section contains the following topics:

- When should I use the Oracle Fusion Middleware System MBean Browser?
- About Reports Configuration MBeans

6.11.1 When should I use the Oracle Fusion Middleware System MBean Browser?

You use the System MBean Browser to enter or modify Oracle Reports configuration settings that are not available in Fusion Middleware Control Oracle Reports pages.

Note: You should not use the System MBean Browser unless you are an advanced middle tier administrator.

6.11.2 About Reports Configuration MBeans

Configuration MBeans are defined for each of the Oracle Reports configuration files. Table 6–2 lists the configuration MBeans for Oracle Reports.

Table 6–2 Reports Configuration MBeans

Configuration MBeans	Associated Configuration File
ServerConfigMXBean	rwserver.conf/rwbuilder.conf
JDBCPDSConfigMXBean	jdbcpds.conf
DiscoveryServiceConfigMXBean	rwnetwork.conf
TextPDSConfigMXBean	textpds.conf
XMLPDSConfigMXBean	xmlpds.conf
CgicmdConfigMXBean	cgicmd.dat
RWServletConfigMXBean	rwservlet.properties
BridgeConfigMXBean	rwbridge.conf
ScreenprinterConfigMXBean	screenprinter.ppd

Note: All Reports environment variables that are set in the registry are not exposed using MBeans. However, if they are specified in the server configuration file ENVID, the environment variables are exposed by `ReportsServerConfigMXBean`. Similarly, all environment variables used in server start/stop shell scripts are not exposed using MBeans.

6.12 Modifying Reports Configuration Settings Using the System MBean Browser

To modify Oracle Reports configuration settings using the System MBean Browser:

1. Start Oracle Enterprise Manager Fusion Middleware Control and display the Farm Home page.
2. Select **Fusion Middleware** > **ReportsDomain** > **WLS_REPORTS**. The WLS_REPORTS page is displayed.
3. From the **WebLogic Server** menu, choose **System MBean Browser**. Fusion Middleware Control displays the System MBean Browser page.
4. Click the plus (+) symbol in the left column to expand a node in the navigation tree and drill down to the MBean you wish to access. The navigation tree expands to display links for viewing or updating settings. Each node in the navigation tree represents settings in a configuration file.
5. Click a node in the navigation tree, and click the **Attributes** tab to display the details for a group of attributes. Attribute details include name, description of each attribute, access details and its current value.
6. Update an attribute value in one of the following ways:
   - Using the current page:
     * Enter a new attribute value into the Value field for the appropriate row.
     * Click **Apply** to apply the changes.
   - Displaying a page to view or update the setting:
     * Click the link in the **Name** column to display a new page.
     * Enter a new attribute value in the **Value** field.
     * Click **Apply** to apply the changes.
     * Click **Return** to close the page and display the navigation tree.
7. Repeat the previous steps to view or update another attribute value.

Note: Only attribute values with write or read-write permissions can be modified.

6.13 Diagnosing Issues

To help diagnose issues, you can use the Oracle Enterprise Manager logging functionality:

- Specifying Logging Information
- Diagnosing Font Issues

6.13.1 Specifying Logging Information

To specify information to be saved in log files for Reports Server or Reports Application (in-process Reports Server):

1. Log in to Oracle Enterprise Manager.
2. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser.
3. Navigate to logging configuration mbean

   Standalone server - `oracle.logging:type=LogConfig,componentType=oracle_repserv,name=<componentName>`

   Inprocess server - `oracle.logging:type=LogConfig,ServerName=<managedServerName>`

   Reports Tools - `oracle.logging:type=LogConfig,componentType=oracle_reptools,name=<componentName>`

   Reports Bridge - `oracle.logging:type=LogConfig,componentType=oracle_repbrd,name=<componentName>`.
4. In **Operations** tab click `updateHandlerProperty` enter handler name and change properties like `path`, `maxLogSize`, `maxFileSize` as desired

   `zrclient_trace_handler` - network trace handler

   `rwengine_trace_handler` - engine trace handler

   `rwserver_trace_handler` - server trace handler

   `rwservlet_trace_handler` - servlet trace handler

   `runtime_trace_handler` - runtime trace handler
5. In Oracle Enterprise Manager, the rotation policy for log files can be set by specifying the Max Log size and Max number of files properties. For example, if Max Log size is set to 10MB and Max number of files is set to 10, log file rotation automatically takes place when the first log file (diagnostics.log) reaches 1 MB (Max Log size / Max number of files = 10MB/10). ODL then renames this file to diagnostic1.log and starts logging to a new diagnostics.log. When it reaches a size of 1 MB, it is renamed to diagnostics2.log and logging continues in diagnostics.log. When the number of files reaches 10, the earliest log file is purged (diagnostics1.log) and a new diagnostics11.log is created. In this example, the maximum size of all log files is limited to 10 MB and the maximum number of files to 10, removing the risk of creating huge log files of arbitrary size and the machine running out of space, bringing the production system down.
6. Restart the component for the changes to take effect.

6.13.2 Diagnosing Font Issues

Refer to Section 6.4.2, "Diagnosing Font Issues".

7

Configuring Oracle Reports Services

When you install Oracle Fusion Middleware, Oracle Reports 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 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:

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

7.1 Oracle Reports Services Configuration Files

This section identifies the configuration files associated with Oracle 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.

Note: Reports application is deployed in no-stage mode in following location:

DOMAIN_HOME/servers/WLS_REPORTS/tmp/_WL_user/reports_*version*/*random_string*

You might see Reports application configuration files in the `configuration` directory in the above specified location, but they are stale files; do not edit them. Any modifications done to the Reports application configuration files through System MBean browser are not reflected in the files located under the configuration directory.

The correct location of Reports application configuration file is:

DOMAIN_HOME/config/fmwconfig/servers/WLS_REPORTS/applications/reports_*version*/configuration

The configuration files associated with Oracle Reports Services are listed and described in Table 7–1.

Note: The paths specified in Table 7–1 are the same for both Windows and UNIX environments, though they are expressed here using the Windows backslash convention (\).

Table 7–1 Oracle Reports Services Configuration Files

Component	Configuration File

Reports Server (`rwserver`)	`ORACLE_HOME\reports\dtd\rwserverconf.xsd`
	For Standalone servers:
	`${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/rwserver.conf`
	For In-process servers:
	`$DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/rwserver.conf`
	The `rwserverconf.xsd` file contains data type definitions for `rwserver.conf` and `rwbuilder.conf` elements and attributes.
	The `rwserver.conf` file defines initial values for the Reports Server Cache, the Oracle Reports engine, and security; registers valid destination types; and sets other server-related values. This file is automatically created when you start up the server.
	See Section 7.2, "Reports Server Configuration File".
	Reports Server network configuration file: `${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/rwnetwork.conf`
	ORACLE_HOME`\reports\dtd\rwnetworkconf.xsd`
	The `rwnetworkconf.xsd` contains data type definitions for `rwnetwork.conf` elements and attributes.
	See Section 7.5, "Network Configuration File".

Table 7–1 (Cont.) Oracle Reports Services Configuration Files

Component	Configuration File
Oracle Reports Builder (rwbuilder) **Oracle Reports Runtime** (rwrun)	*ORACLE_HOME*\reports\dtd\rwserverconf.xsd ${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/rwbuilder.conf The rwserverconf.xsd file contains data type definitions for rwserver.conf and rwbuilder.conf elements and attributes. See Section 7.2.1, "Reports Server Configuration Elements". The rwbuilder.conf file configures the Reports Server that is embedded in Oracle Reports Builder and Oracle Reports Runtime. All run requests must go through Reports Server, meaning that Oracle Reports Builder requires a server to run reports. Oracle Reports Builder automatically starts a Reports Server to handle its requests. When you run a report from Oracle Reports Builder, this file provides the configuration for the in-process Reports Server instance that is invoked. Like the rwserver.conf file, this file relies on the rwserverconf.xsd file for its data type definitions, though the following elements do not apply: persistFile and security. Reports Builder network configuration file: ${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/rwnetwork.conf *ORACLE_HOME*\reports\dtd\rwnetworkconf.xsd The rwnetworkconf.xsd contains data type definitions for rwnetwork.conf elements and attributes. See Section 7.5, "Network Configuration File".
Oracle Reports Servlet (rwservlet)	*ORACLE_HOME*\reports\dtd\rwservlet.xsd $DOMAIN_HOME\config\fmwconfig\servers\<WLS_SERVER_NAME>\applications\reports_<version>\configuration\rwservlet.properties The rwservlet.xsd file contains data type definitions for rwservlet.properties parameters. See Section 7.3, "Oracle Reports Servlet Configuration File". Reports Servlet network configuration file: $DOMAIN_HOME\config\fmwconfig\servers\<WLS_SERVER_NAME>\applications\reports_<version>\configuration\rwnetwork.conf *ORACLE_HOME*\reports\dtd\rwnetworkconf.xsd The rwnetworkconf.xsd contains data type definitions for rwnetwork.conf elements and attributes. See Section 7.5, "Network Configuration File".

Table 7–1 (Cont.) Oracle Reports Services Configuration Files

Component	Configuration File
Oracle Reports Bridge (`rwbridge`)	*ORACLE_HOME*`\reports\dtd\bridgeconfig.xsd` `${DOMAIN_HOME}/config/fmwconfig/components/ReportsBridgeComponent/<reports_bridge_name>/rwbridge.conf` The `bridgeconfig.xsd` file contains data type definitions for `rwbridge.conf` elements and attributes. See Section 7.4, "Oracle Reports Bridge Configuration File". Reports Bridge network configuration file: `${DOMAIN_HOME}/config/fmwconfig/components/ReportsBridgeComponent/<reports_bridge_name>/rwnetwork.conf` Note: The directory, ORACLE_INSTANCE\config\ReportsBridgeComponent is not created by default. Instead, it is created when a user creates a Reports Bridge through `opmnctl`. *ORACLE_HOME*`\reports\dtd\rwnetworkconf.xsd` The `rwnetworkconf.xsd` contains data type definitions for `rwnetwork.conf` elements and attributes. See Section 7.5, "Network Configuration File".

7.1.1 Centralized Reports System Component Configuration

- In 12c reports system component config files are centralized in AdminServer machine
  - `${DOMAIN_HOME}/config/fmwconfig/components/<componentType>/<componentName>/rwserver.conf`
- System components mean - Reports Server, Reports Tools and Reports Bridge
- They are replicated to remote machines if the component is targeted to remote machine. But the central copy will be the source of truth
- In remote machine they are available in same location
- When a change is made to central copy it is replicated to remote machine
  - If change is made via mbeans (recommended), then change is immediately replicated to remote machine when the NodeManager in remote machine is up.
  - If change is made manually, then change is replicated to remote machine when AdminServer is restarted and when the NodeManager in remote machine is up.
  - This means do not make any change to remote copy as they will be overwritten with central copy next time when AdminServer / remote NodeManager starts communicating

7.1.2 Decentralized Reports Application Configuration

- Reports application configuration is not centralized

```
${DOMAIN_
HOME}/config/fmwconfig/servers/<managedServerName>/applications/reports
_<version>/configuration/
```

- They will be local to the machine where the corresponding WLS_REPORTS is targeted
- However some config like logging.xml is a Server level config and is central in nature and replicated as explained above.

```
${DOMAIN_HOME}/config/fmwconfig/servers/<managedServerName>/logging.xml
```

7.2 Reports Server Configuration File

The configuration settings for the Reports Server component of Oracle Reports Services are stored in the XML file `rwserver.conf` and `rwbuilder.conf`, located in the directories specified in Table 7–1.

Both files are supported by the `rwserver.template` file in `ORACLE_HOME\reports\conf`, which contains default server configuration values on both Windows and UNIX.

The `rwserver.conf` file is the default server configuration file. The `rwbuilder.conf` file configures the server instance used in-process by Oracle Reports Builder.

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

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

- The `rwserver.conf` file is created when a new Reports Server component is created.
- The `rwbuilder.conf` file is created when a new Reports Tools component is created.

This section describes:

- Reports Server Configuration Elements
- Dynamic Environment Switching

7.2.1 Reports Server Configuration Elements

The `rwserverconf.xsd` file provides the following data type definitions for configuring `rwserver.conf` and `rwbuilder.conf` elements and attributes:

- ORBPorts
- pluginParam
- cache
- connection
- destination
- environment
- envVariable
- engine
- job
- jobRecovery

- jobStatusRepository
- log
- jobRepository
- notification
- oidconnection
- orbClient
- persistFile
- identifier
- property
- queue
- folderAccess
- security
- proxyServer
- domain
- bypassProxy
- proxyServers
- proxyInfo
- webLayout
- dbProxyKey
- dbProxyConnKeys
- jobThresholds
- server

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

Note that these are XML elements, and XML is case-sensitive. Additionally, when you add any of these elements to the `rwserver.conf` or `rwbuilder.conf` configuration file, you **must follow the order of elements** as described in `rwserverconf.xsd`.

7.2.1.1 ORBPorts

The `ORBPorts` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="ORBPorts">
  <xs:complexType>
    <xs:attribute name="value" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `ORBPorts` element may be specified as shown in this example:

To specify a port range:

```
<ORBPorts value="17000-17010"/>
```

To specify specific ports:

```
<ORBPorts value="17000,17010,17020,17030,17040"/>
```

Required/Optional

Optional. By default, CORBA objects use any available port for communication. Since Reports Server uses CORBA for communication, it will use any available free port for communication. If you want Reports Server to use predefined ports instead of random ports, you must include the `ORBPorts` element in the server configuration file.

Description

The `ORBPorts` element specifies either a range of ports or specific ports for CORBA communication. When `ORBPorts` is specified, Reports Server will choose one of the ports from the list specified for ORB internal communication. One port is needed for Reports Server and one for each engine.

> **Note:** The `ORBPorts` element is used to assign specific ports to Reports Server and engines for running report and other requests. Do not confuse these ports with those you see in Oracle Enterprise Manager through the **Ports** link, which are ports reserved for Reports Server discovery mechanism and the Oracle Reports Bridge component.

You cannot specify port numbers for individual engines. Each engine picks up the next port number in the list. Suppose you have the `maxengine` attribute of the `engine` element set to `5` for `rwEng`, and `URLEng` is also enabled, then you must specify a minimum of 7 ports in the `ORBPorts` element (1 for Reports Server + 5 for `rwEng` + 1 for `rwURLEng`).

The `ORBPorts` element attribute is described in Table 7–2.

Table 7–2 Attribute of the `ORBPorts` Element

Attribute	Valid Values	Description
`value`	Range of values or Numbers separated by commas	The port range that can be used for Reports Server and engine communication through CORBA.

> **Note:** The `ORBPorts` element should be defined only if you have enabled TCP port filtering on your server where Reports Server is running. If port filtering is enabled, you can open few ports for Reports Server, then use `ORBPorts` to specify them in the server configuration file for Reports Server/engine communication. If any of the ports are not available, Reports Server or engines may fail to start and an error displays.

7.2.1.2 pluginParam

The `pluginParam` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="pluginParam">
  <xs:complexType mixed="true">
    <xs:sequence>
```

```
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" use="required" type="xs:ID"/>
    <xs:attribute name="value" use="required" type="xs:string"/>
    <xs:attribute name="type" default="text">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="text"/>
          <xs:enumeration value="file"/>
          <xs:enumeration value="url"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `pluginParam` element may be specified as shown in this example:

```
<pluginParam name="mailServer" value="mail.oracle.com">
  <property name="enableSSL" value="yes"/>
  <property name="mailUserName" value="%MAILUSER%"/>
  <property name="mailPassword" value="%xyz%"/>
</pluginParam>
```

Required/Optional

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

Description

The `pluginParam` element provides a means of specifying plug-ins that can be used by several built-in destinations such as e-mail, JDBC pluggable data source (PDS), Text PDS, and so on. It is *not* used by the FTP and WebDAV built-in destinations, and is not available to custom pluggable destinations, such as fax. Now every server has its own textpds.conf, jdbcpds.conf and xmlpds.conf files.

You can specify any plug-in parameter and name it in any way as long as it is supported or required by the built-in destination.

The `pluginParam` element attributes are described in Table 7–3.

Table 7–3 Attributes of the `pluginParam` Element

Attribute	Valid Values	Description
`name`	string	The name of the plug-in parameter.
Mail Server.		See Properties below for information about specifying the `enableSSL` property when `name="mailServer"`.
`value`.	string	The value of the specified plug-in parameter.

Table 7–3 (Cont.) Attributes of the `pluginParam` Element

Attribute	Valid Values	Description
type	text	**Default:** text
	file	Describes the type of plug-in being specified.
	url	For text, specify the string that is required to identify the named plug-in 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, specify the directory path and filename of the plug-in 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, specify the full, absolute URL required by the plug-in 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 Oracle 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 does not specify a type because the plug-in parameter, a mail server name, is the default type, text.

Properties

You can also optionally enter multiple property sub-elements for the pluginParam element. The only requirement is that they be name/value pairs recognized by the specified plug-in parameter. For example:

```
<pluginParam name="mailServer" value="%MAILSERVER%">
  <property name="enableSSL" value="yes"/>
</pluginParam>
```

In this example, the property sub-element specifies the enableSSL property, which is only applicable to mailServer. If the specified mailServer is SSL-enabled, it rejects plain connection requests, so it is necessary to use SSL Sockets to establish a connection with the specified mailServer and send emails, by default, the value of enableSSL is no for compatibility with prior releases.

7.2.1.3 cache

The cache element is defined in rwserverconf.xsd as follows:

```
<xs:element name="cache">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="class" default="oracle.reports.cache.RWCache"
                  type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In rwserver.conf, the cache element may be specified as shown in this 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 specifies 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 Oracle Reports Services Cache API.

> **Note:** For more information about the cache API, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html).

The cache element attribute is described in Table 7–4.

Table 7–4 Attribute 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.

Properties

You can also optionally enter multiple property sub-elements for the cache element. The only requirement is that they be name/value pairs recognized by the implementation class you register under cache. For example, if you use the default cache Java class that is provided with Oracle 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="home/john/HRInstance/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"/>
```

Maximum Cached Files.

- `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"/>
```

7.2.1.4 connection

The `connection` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="connection">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="orbClient" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>

    <xs:attribute name="idleTimeOut" default="15">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="maxConnect" default="50">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `connection` element may be specified as shown in this example:

```
<connection idleTimeOut="20" maxConnect="50" >
  <orbClient id="RWClient" publicKeyFile="clientpub.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 7–5). 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.

The `connection` element attributes are described in Table 7–5.

Table 7–5 Attributes of the `connection` Element

Attribute	Valid Values	Description
`idleTimeOut`	Number	**Default:** `15`
Connection Idle Timeout (min).		Allowable amount of time in minutes the connection can be idle.
`maxConnect`	Number	**Default:** `50`
Maximum Connections.		The maximum number of requests that Reports Server can service simultaneously. Requests in excess of the `maxConnect` value return a Java exception.

The `connection` element also includes the `orbClient` sub-element, described in Section 7.2.1.16, "orbClient".

7.2.1.5 destination

The `destination` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="destination">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="class" use="required" type="xs:string"/>
    <xs:attribute name="destype" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `destination` element may be specified as shown in this example:

```
<destination destype="oraclePortal" class="oracle.reports.server.DesOraclePortal">
  <property name="portalUserid"
            value="portal_db_username/portal_password@portal_db_connection"
            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, e-mail, file, cache, and Oracle 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.

You need not register the following default destinations:

- Cache
- E-mail
- Printer
- File

- FTP
- WebDAV

You *may* want to register the following default destination:

- Oracle Portal: The entry for this destination is created by default in the server configuration file, but it is commented out. To start using this destination, you must uncomment the `destination` entry, and also provide appropriate property values (for example, the value for the `portalUserid` property).

You must register any new destination types you create through the Oracle Reports Services Destinations API.

> **Note:** For more information about the `destination` API, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html).
>
> Configuring destinations is discussed in detail in Chapter 13, "Configuring Destinations for Oracle Reports Services".

The `destination` element attributes are described in Table 7–6.

Table 7–6 Attributes of the `destination` Element

Attribute	Valid Values	Description

`class`	string	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`

`destype`	string	Identifies the destination type; for example: `destype="printer"`

Properties

You can also optionally enter multiple `property` sub-elements for 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="dbuser" value="$$PORTAL_DB_USERNAME$$"/>
     <property name="dbpassword" value="csf:$$CSF_ALIAS$$:$$PORTAL_DB_PASSWORD_
KEY$$"/>
     <property name="dbconn" value="$$PORTAL_DB_TNSNAME$$"/>
   </destination>
```

In this example, the `property` sub-element provides connect information to enable Reports Server to access Oracle Portal. The `encrypted` attribute is 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 (`scott/tiger@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.

7.2.1.6 environment

The `environment` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="environment">
  <xs:complexType>
     <xs:sequence>
        <xs:element ref="envVariable" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="id" use="required" type="xs:ID"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `environment` element may be specified as shown in this 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 (that is, environment variables) that you want to use to establish a particular runtime environment. You may include as many environment elements as you need (for example, one for each language/territory you must 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 Section 7.2.1.8, "engine".
- The command line keyword, `ENVID`, of your report's job request, which makes the environment settings only effective for that particular report job request.

The `environment` element attribute is described in Table 7–7.

Table 7–7 Attribute of the `environment` Element

Attribute	Valid Values	Description
`id`	string	The name of the environment. **Default Env ID**.

The environment element includes one or more envVariable sub-elements, described in Section 7.2.1.7, "envVariable".

7.2.1.7 envVariable

The envVariable element is defined in rwserverconf.xsd as follows:

```
<xs:element name="envVariable">
  <xs:complexType>
     <xs:attribute name="name" use="required" type="xs:string"/>
     <xs:attribute name="value" use="optional" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In rwserver.conf, the envVariable element may be specified as shown in this example:

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

Required/Optional

Optional.

Description

Each envVariable is specified as a name–value pair. They can be either standard environment variables or user-defined environment variables.

envVariable is a sub-element of the environment element.

The envVariable element attributes are described in Table 7–8.

Table 7–8 Attributes of the envVariable Element

Attribute	Valid Values	Description
name **Add**.	string	The name of the environment you wish to use (for example, NLS_LANG).
value **Add**.	string	The value you want to assign to the environment variable identified with the name attribute (for example, Japanese_Japan.JA16SJIS).

7.2.1.8 engine

The engine element is defined in rwserverconf.xsd as follows:

```
<xs:element name="engine">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>

    <xs:attribute name="id" use="required" type="xs:string"/>
    <xs:attribute name="class" use="required" type="xs:string"/>

    <xs:attribute name="maxEngine"  use="required">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
```

```
            <xs:minInclusive value="1"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="minEngine" use="required">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="engLife"  use="required">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
        <xs:minInclusive value="1"/>
        </xs:restriction>
       </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="maxIdle" default="30">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="callbackTimeOut"  default="90000">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="60000"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="engineResponseTimeOut">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="initEngine" type="xs:integer"/>
    <xs:attribute name="jvmOptions" type="xs:string"/>
    <xs:attribute name="classPath" type="xs:string"/>
    <xs:attribute name="defaultEnvId" type="xs:string"/>

  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `engine` element may be specified as shown in this example:

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="1"
    maxEngine="5" minEngine="1" engLife="50" maxIdle="15" callbackTimeOut="90000">
  <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.

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 engines provided with Oracle Reports Services or develop your own implementation through the Oracle 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:** For more information about the `engine` API, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html).

The `engine` element attributes are described in Table 7–9.

Table 7–9 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`	string	**Default:** `oracle.reports.engine.EngineImpl` A fully qualified Java class that implements two interfaces: `oracle.reports.engine.Engine` and `oracle.reports.engine.EngineInterface`.
`maxEngine` **Maximum Engines**.	number	**Default:** `1` The maximum number of this type of engine that can run on the server.
`minEngine` **Minimum Engines**.	number	**Default:** `0` The minimum number of this type of engine that is maintained by the server.
`engLife` **Maximum Job Before Restart**.	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.

Table 7–9 (Cont.) Attributes of the `engine` Element

Attribute	Valid Values	Description
`maxIdle` **Maximum Idle Before Shutdown (min)**.	number	**Default:** `30` The number of minutes of allowable idle time before the engine is shut down However, the current number of engines should be 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:** `90000` 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.
`engineResponseTimeOut` **Engine Response Timeout (min)**.	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 (for example, due to the engine hanging or a long blocking SQL query), Reports Server terminates the job.
`initEngine`	number	**Default:** `1` The number of engines you want Reports Server to start at initialization. When running a report using `rwrun`, retain the default value of `initEngine="1"`. Because `rwrun` can run only one report at a time, any other setting may result in the report not being run.
`jvmOptions` **JVM Options**.	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. If this attribute is not specified, the engine running in the server environment uses the JVM options specified by the value of the `REPORTS_JVM_OPTIONS` environment variable. See Section B.1.55, "REPORTS_JVM_OPTIONS".

`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"`

Table 7–9 (Cont.) Attributes of the `engine` Element

Attribute	Valid Values	Description
`defaultEnvId` **Default Env ID**.	string	(Optional attribute) The default environment within which Reports Server starts an engine. The attribute takes an ID associated with an `environment` element in the server configuration file. When `defaultEnvId` is specified, 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 `defaultEnvId` is not specified, Reports Server spawns engines with the environment settings in force at startup time. For more information, refer to Section 7.2.2, "Dynamic Environment Switching".

Properties

You can also optionally enter multiple `property` sub-elements for the `engine` element. The only requirement is that they be name/value pairs recognized by the Java class that implements the Oracle Reports engine.

Table 7–10 Properties of the `engine` Element

Property	Valid Values	Description
`sourceDir` **Reports Source Directory**.	directory path	The default directory you will use for report definition files. It overrides path information specified in the `REPORTS_PATH` environment variable. The directory specified by `sourcedir` is not given access by default. It used only to search for files, and Oracle Reports handles any associated security settings separately if and when the target files are located. See the example that follows this table.
`tempDir` **Reports Temp Directory**.	directory path	The name and location of the temporary directory Oracle Reports Services will use for its temporary files. If this value is unspecified for a default engine, Oracle Reports Services uses the temporary directory specified in the `REPORTS_TMP` environment variable. If `REPORTS_TMP` is not specified, Oracle Reports Services uses your operating system's default temporary directory. See the example that follows this table.

Table 7–10 (Cont.) Properties of the `engine` Element

Property	Valid Values	Description
`keepConnection` **Keep Database Connection**.	`YES`|`NO`	**Default:** `YES` Used by the default runtime engine implementation (that is, `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 `rwserver.conf` file used in previous releases (for example, 9.0.2.x) runs in the current environment.
`diagnosis` **Enable Engine Diagnostics**.	`YES`|`NO`	Diagnoses whether or not a specific function in a report run completed successfully. The diagnostic log provides information on important checkpoints or tasks in the engine during a report run. This information is useful in cases where the engine stops responding, resulting in "hanging" jobs. `YES` Diagnostic information is written to the `diagnostic.log` file. The engine diagnosis option provides more detailed information than report tracing, which is typically used to debug the execution of a report to provide information such as the file currently formatting, or report trigger currently running. See the example that follows this table.

Example of `sourcedir` and `tempDir` properties: If you use the default `engine` Java class that is provided with Oracle Reports Services, your `engine` configuration entry might look like this (in a Windows environment):

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="1"
    maxEngine="5" minEngine="1" engLife="50" maxIdle="15" callbackTimeOut="90000">
  <property name="sourceDir" value="D:\orawin\reports\myReport"/>
  <property name="tempDir" value="D:\orawin\reports\myTemp"/>
</engine>
```

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

Example of `diagnosis` property: To enable the engine diagnosis option, your `engine` configuration element might look like this:

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="2"
    maxEngine="8" minEngine="1" engLife="1" maxIdle="3" callbackTimeOut="90000">
  <property name="diagnosis" value="yes"/>
</engine>
```

7.2.1.9 job

The `job` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="job">
  <xs:complexType>
    <xs:attribute name="engineId" use="required" type="xs:string"/>
    <xs:attribute name="jobType" default="report" type="xs:string"/>
    <xs:attribute name="securityId" type="xs:string"/>
    <xs:attribute name="retry" default="0">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:minInclusive value="0"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `job` element may be specified as shown in this example:

```
<job jobType="report" engineId="rwEng" securityId="rwSec" retry="3"/>
```

Required/Optional

Required. You must have at least one `job` element.

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 Oracle 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.

The `job` element attributes are described in Table 7–11.

Table 7–11 Attributes of the `job` Element

Attribute	Valid Values	Description
`engineId`	string	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.

Table 7–11 (Cont.) 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. The database authentication functionality provided in Oracle Reports is available only when `jobType=report`. This is the job type of the default engine (`rwEng`) provided with Oracle Reports Services. The database authentication functionality is not implemented when `jobType` specifies a different value (for example, for a custom engine that you develop yourself). This is because a custom engine may require a different format for the connect string, while the Oracle Reports database authentication functionality limits the connect string to the Oracle Reports format `user/password@dbname` used for the default engine.
`securityId`	string	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.
`retry` **Job Retries**.	integer	**Default:** `0` When `jobType="report"`, specifies the number of times to retry a job after the initial run, if the job fails due to an engine crash or unexpected error. The job is resubmitted to another engine the number of times specified. This attribute is ignored if the job is explicitly cancelled or when `jobType="rwurl"` (URL engine). If an invalid value is specified, this attribute is ignored and the default value of `0` is used. If JOBRETRY is specified on the command line, it takes precedence, and the `retry` attribute is ignored.

7.2.1.10 jobRecovery

The `jobRecovery` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="jobRecovery">
  <xs:complexType>
    <xs:attribute name="auxDatFiles" default="no">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="yes"/>
          <xs:enumeration value="no"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `jobRecovery` element may be specified as shown in this example:

```
<jobRecovery auxDatFiles="yes"/>
```

Required/Optional

Optional. To enable the job recovery mechanism, add the jobRecovery element to the server configuration file. The job recovery mechanism is disabled by default.

Description

The jobRecovery element includes the auxDatFiles attribute. When auxDatFiles=yes, Oracle Reports enables a more resilient job recovery mechanism for maximal retrieval of jobs in case the original .dat file is corrupt due to some reason. When auxDatFiles=yes, Reports Server creates the following two auxiliary files in addition to *server_name*.dat (the main .dat file):

- *datfilename*_offset.dat contains the auxiliary information of jobs in the main.dat file, which helps in retrieving jobs from the main.dat file.
- *datfilename*_sc.dat contains all scheduled jobs information (in addition to the information stored in main.dat file).

If the job recovery mechanism is enabled, Reports Server on startup reads the main.dat file with the help of the *datfilename*_offset.dat file using the auxiliary information stored in it. If the main.dat file is corrupt and Reports Server cannot retrieve all the jobs information, it starts reading the *datfilename*_sc.dat file and recovers the scheduled jobs for this file. Thus, *datfilename*_sc.dat serves as a backup file, which results in maximum possibility of recovery of scheduled jobs in case of corruption of the main.dat file.

If Reports Server fails to find the *datfilename*_offset.dat file (for example, when the jobRecovery element is enabled for first time) when the job recovery mechanism is enabled, it reads the jobs from the main.dat file and creates the other two auxiliary files from scratch.

The *server_name*.dat, *datfilename*_offset.dat, and *datfilename*_sc.dat files form a unique triplet, and the auxiliary files are valid only when the job recovery mechanism is enabled. If the auxiliary files are found when the job recovery mechanism is disabled, Reports Server deletes these files from the file system to maintain the integrity between these files. For this reason, you must always handle these three files together (for example, if you are copying a file from one machine to another, you must copy these three files together).

The jobRecovery element attribute is described in Table 7–12.

Table 7–12 Attribute of the jobRecovery Element

Attribute	Valid Values	Description
auxDatFiles	yes	**Default:** no
	no	See above.

7.2.1.11 jobStatusRepository

The jobStatusRepository element is defined in rwserverconf.xsd as follows:

```
<xs:element name="jobStatusRepository">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="class" default="oracle.reports.server.JobRepositoryDB"
                  type="xs:string"/>
```

```
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `jobStatusReposity` element may be specified as shown in this example:

```
<jobStatusRepository>
<property name="dbuser" value="<dbuser>"/>
<property name="dbpassword" value=csf:reports:"<dbkey>"/>
<property name="dbconn" value="<dbconn>"/>
</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` element 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.

The `jobStatusRepository` element attribute is described in Table 7–13.

Table 7–13 Attribute of the `jobStatusRepository` Element

Attribute	Valid Values	Description
`class` **Enable Job Repository DB**.	string	**Default:** `oracle.reports.server.JobRepositoryDB` A fully qualified Java class that implements the Reports Server Job Repository Java class (`oracle.reports.server.JobRepository`).

Properties

You can also optionally enter multiple `property` sub-elements for the `jobStatusRepository` element for passing options into the repository. The only requirement is that they be name/value pairs recognized by the class you specify in the server configuration file.

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

```
<jobStatusRepository>
<property name="dbuser" value="<dbuser>"/>
<property name="dbpassword" value=csf:reports:"<dbkey>"/>
<property name="dbconn" value="<dbconn>"/>
</jobStatusRepository>
```

Note: Oracle Reports uses the `dbconn` property of the `jobstatusrepository` element to connect to the database when updating the log information about job queues.

7.2.1.12 log

The `log` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="log">
  <xs:complexType>
    <xs:attribute name="option" default="noJob">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="allJobs"/>
          <xs:enumeration value="succeededJobs"/>
          <xs:enumeration value="failedJobs"/>
          <xs:enumeration value="noJob"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `log` element may be specified as shown in this 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_INSTANCE`*`/diagnostics/logs/ReportsSeverComponent/<reports_server_name>/rwserver_diagnostic.log`

The `log` element attribute is described in Table 7–14.

Table 7–14 Attribute of the `log` Element

Attribute	Valid Values	Description

`option`	`allJobs`
	`succeededJobs`
	`failedJobs`
	`noJob`
		Default: `noJob`
		Describes the type of jobs that are logged. This is in addition to the default server activities that are logged. Choose from the following options:
		■ `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

7.2.1.13 jobRepository

The `jobRepository` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="jobRepository">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `jobRepository` element may be specified as shown in this example:

```
<jobRepository>
<property name="dbuser" value="dbuser"/>
<property name="dbpassword" value="csf:reports:dbpasswdKey"/>
<property name="dbconn" value="dbconn"/>
</jobRepository>
```

Required/Optional

Required in a high availability (HA) environment. Optional in a non-HA environment. You can have a maximum of one `jobRepository` element in your server configuration file.

Description

The `jobRepository` element enables you to store all job information in the database instead of the file system (that is, in DAT files). This element is mandatory if you want to use high availability (HA), because Reports Servers in the group share job information, which is possible only if the job information is stored in the database, and not individual DAT files.

The `jobRepository` element has no attributes.

Properties

jobRepository requires only one property sub-element, `repositoryconn`. and the jobRepository element enables you to store all job information in the database or the file system (that is, in DAT files).

7.2.1.14 notification

The `notification` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="notification">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="class" use="required" type="xs:string"/>
    <xs:attribute name="id" default="mailNotify" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `notification` element may be specified as shown in this 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 through e-mail, or design your own with the Oracle Reports Notification API.

> **Note:** For more information about the `notification` API, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html).

The `notification` element attributes are described in Table 7–15.

Table 7–15 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 command line keyword `notifysuccess` and `notifyfailure` to specify the email address where notification should be sent (see Appendix A, "Command-Line Keywords"). 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/>
```

The succNoteFile and failNoteFile **Email notification file for success** and **Email notification file for failure**.

Some mail servers may validate the sender's domain name. If the notification fails because of this domain name validation, then you must add the following property as part of the notification element:

```
<property name="sender" value="valid email address"/>
```

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.

7.2.1.15 oidconnection

The oidconnection element is defined in rwserverconf.xsd as follows:

```
<xs:element name="oidconnection">
  <xs:complexType>
    <xs:attribute name="increment" default="10" type="xs:integer"/>
    <xs:attribute name="init" default="10" type="xs:integer"/>
    <xs:attribute name="timeout" default="0" type="xs:integer"/>
  </xs:complexType>
</xs:element>
```

Example

In rwserver.conf, the oidconnection element may be specified as shown in this example:

```
<oidconnection init="10" increment="10" timeout="600"/>
```

Required/Optional

Optional.

Description

The oidconnection element specifies Oracle Internet Directory connection pooling parameters for Reports Server. In a production environment, you can use this parameter to provide granular control over Oracle Internet Directory connection pooling of Reports Server, namely:

- The number of connections to keep open in the pool when Reports Server is initialized.
- Upon exhausting the available connections, the number of new connections to be added to the pool when a new request arrives.
- The timeout for closing idle open Oracle Internet Directory connections to reduce the resource usage.

The oidconnection element attributes are described in Table 7–16.

Table 7–16 Attributes of the oidconnection Element

Attribute	Valid Values	Description
init	number	**Default:** "10" Initial number of Oracle Internet Directory connections to be created when Reports Server is initialized.

Table 7–16 (Cont.) Attributes of the `oidconnection` Element

Attribute	Valid Values	Description
`increment`	number	**Default:** `"10"` Number of connections to be incremented when all connections are used up.
`timeout`	number	**Default:** `"0" (which specifies no timeout)` Time in seconds for which a connection can be idle before it is closed.

Note: Setting much lower or higher values than the default values for these attributes can have a performance impact on Oracle Reports Services. In a typical production environment, the default values are recommended.

For Oracle Reports Servlet (`rwservlet`), you can specify Oracle Internet Directory connection pooling parameters using the oidconnection element in the `rwservlet.properties` file.

7.2.1.16 orbClient

The `orbClient` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="orbClient">
  <xs:complexType>
    <xs:attribute name="id" use="required" type="xs:string"/>
    <xs:attribute name="publicKeyFile" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `orbClient` element may be specified as shown in this example:

```
<orbClient id="RWClient" publicKeyFile="clientpub.key"/>
```

Required/Optional

Optional. If you do not specify the orbClient element in your server configuration file, the default values will be used.(see Table 7–17)

Description

The `orbClient` element specifies 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` elements in your server configuration file.

`orbClient` is a sub-element of the connection element

The `orbClient` element attributes are described in Table 7–17.

Table 7–17 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.
`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.

Oracle 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 Oracle 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:

On Windows:

`rwgenkey.bat` *`path_and_client_public_key_file_name path_and_client_private_key_file_name`*

On 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.

7.2.1.17 persistFile

The `persistFile` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="persistFile">
  <xs:complexType>
    <xs:attribute name="fileName" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `persistFile` element may be specified as shown in this 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 attribute is described in Table 7–18.

Table 7–18 Attribute 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_INSTANCE*\reports\server\ The path is the same for Windows or UNIX.

7.2.1.18 identifier

The identifier element is defined in rwserverconf.xsd as follows:

```
<xs:element name="identifier">
  <xs:complexType mixed="true">
    <xs:attribute name="encrypted" default="no">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="yes"/>
          <xs:enumeration value="no"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

In rwserver.conf, the identifier element may be specified as shown in this example:

```
<identifier 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 by the Reports Configuration Assistant when you first install Oracle Reports. The Reports Configuration Assistant sets the values in the form SERVERACCESSKEY/12312312313, where SERVERACESSKEY is the user name and the random generated number (12312312313) is the password. This user name and password is then encrypted and written to rwserver.template and targets.xml during the time of configuring Oracle

Reports Services. Any Reports Server started after the installation will have this identifier information stored in its configuration file.

For a non-secured Reports Server, the values of the `identifier` element is used when:

- Connecting to a Reports Server through the Reports Queue Manager.
- Shutting down a Reports Server through the command line.

In either of these cases, you must provide the `authid` in the command line that matches the values specified in the `identifier` element. To provide a specific password (as the password is a pseudo random number), you must do the following:

1. Edit the server configuration file, `rwserver.conf`.
2. Replace the encrypted `username/password` values generated with custom values.
3. Set `encrypted=no`.

   For example:

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

4. Restart Reports Server. Reports Server sets `encrypted=yes` when it restarts.
5. Edit the `targets.xml` file and specify the same `username` and `password` values that were included in the `rwserver.conf` file.

You should restart Reports Server, immediately, after making this change. Reports Server automatically encrypts the user name and password and resets `encrypted` to `yes`. The values should now read as follows:

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

For a secure Reports Server, the authentication is done by the security infrastructure; that is, by using the Oracle Internet Directory repository. Thus, you cannot pass the values in the `identifier` element to shut down a Reports Server or launch Reports Queue Manager through the console window.

> **Note:** This user name and password is also used for accessing 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 Section 7.3, "Oracle Reports Servlet Configuration File".

7.2.1.19 property

The `property` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="property">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute name="value" use="required" type="xs:string"/>
    <xs:attribute name="encrypted" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

See the following element descriptions for information about specifying the property element in rwserver.conf:

- Section 7.2.1.3, "cache"
- Section 7.2.1.5, "destination"
- Section 7.2.1.8, "engine"
- Section 7.2.1.11, "jobStatusRepository"

7.2.1.20 queue

The queue element is defined in rwserverconf.xsd as follows:

```
<xs:element name="queue">
  <xs:complexType>
    <xs:attribute name="maxQueueSize" default="1000">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="100"/>
          </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

In rwserver.conf, the queue element may be specified as shown in this 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 maxQueueSize, 1000, will remain in effect.

Description

Use the queue element to specify the maximum number of jobs that can be held in a completed job queue. Oracle 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.By default reports server queue size is 1000 jobs.

If you increase the queue size to more than 3000, and use Reports Queue Manager (rwrqm.exe) to monitor the queue, Queue Manager may fail. When a queue size of 3000 or greater is required, use Oracle Reports Servlet (rwservlet) to manage and monitor the Reports Server jobs queue.

The queue element attribute is described in Table 7–19.

Table 7–19 Attribute of the queue Element

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

7.2.1.21 folderAccess

The folderAccess element is defined in rwserverconf.xsd as follows:

```
<xs:element name="folderAccess">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="read" type="xs:string"/>
      <xs:element name="write" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example

In rwserver.conf, the folderAccess element may be specified as shown in following two examples:

```
<folderAccess>
 <read>c:\myreports\*;C:\Samples\*</read>
   <write>c:\myoutputs\*</write>
</folderAccess>
```

> **Note:** In the above example, the Samples should be downloaded from OTN location http://www.oracle.com/technetwork/middleware/reports/downloads/index.html and then be placed at local location C:\Samples.

The next example specifies access to only the mentioned level folder and sub-folders:

```
<folderAccess>
 <read>c:\myreports;C:\Samples\Subsamples</read>
   <write>c:\myoutputs</write>
</folderAccess>
```

Required/Optional

Optional.

Description

The folderAccess element defines read and write access to file system folders for both secured and non-secured Reports Server, Reports Application (in-process Reports Server), or Oracle Reports Runtime.

The folderAccess element has no attributes. It includes two sub-elements:

- read: specifies the folder(s) to which the Reports Server, Reports Application (in-process Reports Server), or Oracle Reports Runtime has read access only. **Enable File System Access Control** > **Read Directories**.
- write: specifies the folder(s) to which the Reports Server can write. **Enable File System Access Control** > **Write Directories**.

In the example above, the report definition files located in `c:\myreports` and `C:\Samples` are allowed to run only. Similarly, when `destype=file`, the output file can be created only in `c:\myoutputs` (`desname=c:\myoutput\test.pdf`).

Note: Blank or * in the `read` or `write` sub-element are not allowed. A folder needs to be defined for the `<read>` or `<write>` sub-elements and when using it in combination of "*" then any sub-directory under `/folder/` can be used. Separate the directory names with a semicolon (;).

7.2.1.22 security

The `security` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="security">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="class" use="required" type="xs:string"/>
    <xs:attribute name="id" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `security` element may be specified as shown in this example for 11*g*:

```
<security id="rwJaznSec" class="oracle.reports.server.RWJAZNSecurity"/>
```

For backward compatibility, the `security` element may be specified as:

```
<security id="rwSec" class="oracle.reports.server.RWSecurity">
<property name="oidAppEntity" value="$$Self.oidAppEntity$$"/>
<property name="oidUrl" value="$$Self.oidUrl$$"/>
<property name="oidPasswdKey" value="$$Self.oidPasswdKey$$"/>
<property name="portalUserName" value="$$Self.portalUserName$$"/>
<property name="portalConnection" value="$$Self.portalConnection$$"/>
<property name="portalPasswdKey" value="$$Self.portalPasswdKey$$"/>
</security
```

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 Oracle Reports Services, or develop your own implementation through the Reports Server Security API.

> **Note:** For more information about the `security` API, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (`http://www.oracle.com/technetwork/middleware/reports/overview/index.html`).

The `security` element attributes are described in Table 7–20.

Table 7–20 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 for 11*g*:
		`id="rwJaznSec"`
		For backward compatibility,
		`id="rwSec"`

`class`	See the **Description** column	**Default for 11g:** `oracle.reports.server.RWJAZNSecurity` **Default for backward compatibility:** `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 Oracle Portal.

You can associate multiple properties with the `security` element. The only requirement is that they be name/value pairs recognized by the Java class that implements Reports Server security.

The value of all the properties is set by the Installer upon installation. Reports Server uses this entity to connect to Oracle Internet Directory and Portal. Components of the Oracle Fusion Middleware can all connect to Oracle Internet Directory and Oracle Portal, but each component may have different privileges in the directory. Hence, each component needs to identify itself through its own entity name to Oracle Internet Directory when it connects. The Oracle Reports Services entity is of the following format:

`reportsApp_`*`hostname`*`_GUID`

For example:

```
reportsApp_testhost.mydomain.com_BBEFDCDAC2343600E0340800020C7BBCC
```

7.2.1.23 proxyServer

The `proxyServer` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="proxyServer">
 <xs:complexType>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="port" type="xs:string" use="required"/>
  <xs:attribute name="protocol" default="all">
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value="http"/>
     <xs:enumeration value="https"/>
     <xs:enumeration value="ftp"/>
     <xs:enumeration value="file"/>
     <xs:enumeration value="all"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
 </xs:complexType>
```

```
</xs:element>
```

Example

In rwserver.conf, the proxyServer element may be specified as shown in this example:

```
<proxyServer name="www-proxy.example.com" port="80" protocol="all"/>
```

Required/Optional

Optional.

Description

Element that specifies the name, port and protocol of proxy server to be used in order to connect to external network.

The proxyServer element attributes are described in Table 7–20.

Table 7–21 Attributes of the proxyServer Element

Attribute	Valid Values	Description
name	string	the host name of the proxyserver. example.
port	string	the port where the proxyserver is listening on.
protocol	http/https/ftp/file/ all	the protocol used by the proxyserver.

7.2.1.24 domain

The domain element is defined in rwserverconf.xsd as follows:

```
<xs:element name="domain">
 <xs:complexType mixed="true">
 </xs:complexType>
</xs:element>
```

Example

In rwserver.conf, the domain element may be specified as shown in this example:

```
<bypassProxy>
<domain><localhost</domain>
<domain>127.0.0.1</domain>
</bypassProxy>
```

Required/Optional

Optional.

Description

Element that specifies the name of the proxy server for which proxy setting should not be used. The domain element has no attributes.

7.2.1.25 bypassProxy

The bypassProxy element is defined in rwserverconf.xsd as follows:

```
<xs:element name="bypassProxy">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="domain" minOccurs="0" maxOccurs="unbounded"/>
```

```
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `bypassProxy` element may be specified as shown in this example:

```
<bypassProxy>
     <domain>localhost</domain>
     <domain>127.0.0.1</domain>
</bypassProxy>
```

Required/Optional

Optional.

Description

The element that provides a list of domains which specifies name of the proxy server for which proxy setting should not be used

The `bypassProxy` element has no attributes. It includes the `domain` sub-element (see Section 7.2.1.24, "domain").

7.2.1.26 proxyServers

The `proxyServers` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="proxyServers">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="proxyServer" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `proxyServers` element may be specified as shown in this example:

```
<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>
```

Required/Optional

Optional.

Description

Element that specifies a list of proxy servers used by reports server.

The `proxyServers` element has no attributes. It includes the `proxyServer` sub-element (see Section 7.2.1.23, "proxyServer").

7.2.1.27 proxyInfo

The `proxyInfo` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="proxyInfo">
```

```
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="proxyServers"/>
   <xs:element ref="bypassProxy"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `proxyInfo` element may be specified as shown in this example:

```
<proxyInfo>
  <proxyServers>
    <proxyServer name="www-proxy.example.com" port="80" protocol="all"/>
  </proxyServers>
  <bypassProxy>
    <domain>localhost</domain>
    <domain>127.0.0.1</domain>
  </bypassProxy>
</proxyInfo>
```

Required/Optional

Optional.

Description

Element specifying proxy servers used by reports and the bypass hosts for which proxy should not be used.

The `proxyInfo` element has no attributes. It includes two sub-elements:

- `proxyServers`: see Section 7.2.1.26, "proxyServers"
- `bypassProxy`: see Section 7.2.1.25, "bypassProxy"

7.2.1.28 webLayout

The `webLayout` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="webLayout">
  <xs:complexType>
    <xs:attribute name="port" type="xs:string"/>
    <xs:attribute name="docroot" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `webLayout` element may be specified as shown in this example:

```
<webLayout port="8888" docroot="$DOMAIN_HOME/servers/WLS_REPORTS/tmp/_WL_
user/reports_<version>/<random_string>/war"/>
```

Required/Optional

Optional.

Description

`webLayout` element is required to run a report to weblayout using Reports Builder.

The `webLayout` element attributes are described in Table 7–22.

Table 7–22 Attributes of the `webLayout` Element

Attribute	Valid Values	Description
`port`	string	the OHS main port. If OHS is not present, it should be reports managed server port.
`docroot`	string	the location of the web.war where Reports Application is deployed

7.2.1.29 dbProxyKey

The `dbProxyKey` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="dbProxyKey">
 <xs:complexType>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="database" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `dbProxyKey` element may be specified as shown in this example:

```
<dbProxyConnKeys>
  <dbProxyKey name=key1 database=db1/>
  <dbProxyKey name=key2 database=db2/>
</dbroxyConnKeys>
```

Required/Optional

Optional.

Description

The `dbProxyKey` consists of the name and database parameters. It is obtained from the server configuration file based on the database mentioned in the `userid` commandline parameter.

The `dbProxyKey` element attributes are described in Table 7–23.

Table 7–23 Attributes of the `dbProxyKey` Element

Attribute	Valid Values	Description
`name`	string	the name of the key.
`database`	string	the name of the database.

7.2.1.30 dbProxyConnKeys

The `dbProxyConnKeys` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="dbProxyConnKeys">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="dbProxyKey" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `dbProxyConnKeys` element may be specified as shown in this example:

```
<dbProxyConnKeys>
<dbProxyKey name=key1 database=db1/>
<dbProxyKey name=key2 database=db2/>
</dbProxyConnKeys>
```

Required/Optional

Optional.

Description

The `dbProxyConnKeys` element has no attributes. It includes the `dbProxyKey` sub-element (see Section 7.2.1.29, "dbProxyKey").

7.2.1.31 urlEngineAccess

The `urlEngineAccess` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="urlEngineAccess">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="urlPatterns" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
  </xs:complexType>

 <xs:element name="urlPatterns">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="urlPattern" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
```

Example

```
<urlEngineAccess>
     <urlPatterns>
      <urlPattern>http://host:port/myservlet/*</urlPattern>
      <urlPattern>http://host:port/servlet2/myreportsOnly/*</urlPattern>
     </urlPatterns>
   </urlEngineAccess>
```

Required/Optional

Optional

Description

This element specifies patterns of allowed URLs which can be accessed by reports URL engine. By default, the list of URL patterns is empty, that is, no URL is allowed. Note that only HTTP and HTTPS protocols are allowed.

7.2.1.32 jobThresholds

The `jobThresholds` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="jobThresholds">
 <xs:complexType>
  <xs:attribute name="longRunning" default="180">
```

```
   <xs:simpleType>
    <xs:restriction base="xs:integer">
     <xs:minInclusive value="1"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="potentialRunAway" default="180">
   <xs:simpleType>
    <xs:restriction base="xs:integer">
     <xs:minInclusive value="1"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
 </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `jobThresholds` element may be specified as shown in this example:

```
<jobThresholds longRunning="180" potentialRunAway="180"/>
```

Required/Optional

Optional.

Description

jobThreshold consists of two attributes, `longRunning` and `PotentialRunAway`. See Table 7–24.

Table 7–24 Attributes of the `jobThresholds` Element

Attribute	Valid Values	Description
`longRunning`	seconds	the cut-off time for a job after which it is considered a long run job.
		a job that takes relatively longer time.
`potentialRunAway`	seconds	the cut-off time for a currently running job after which it is considered as a potential runaway job.
		a job which has relatively lesser chance of successful completion.

7.2.1.33 server

The `server` element is defined in `rwserverconf.xsd` as follows:

```
<xs:element name="server">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="cache" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="engine" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="environment" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="security" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="oidconnection" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="destination" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="job" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="notification" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="log" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="jobStatusRepository" minOccurs="0" maxOccurs="1"/>
```

```
      <xs:element ref="jobRepository" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="trace" minOccurs="0" maxOccurs="1"/-->
      <xs:element ref="connection" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="ORBPorts" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="queue" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="folderAccess" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="persistFile" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="jobRecovery" minOccurs="0" maxOccurs="1"/-->
      <xs:element ref="identifier" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="proxyInfo" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="pluginParam" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="webLayout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="dbProxyConnKeys" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="jobThresholds" minOccurs="0" maxOccurs="1"/>
<xs:element ref="urlEngineAccess" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In `rwserver.conf`, the `server` element may be specified as shown in this example:

```
<server>
  one or more element specifications
</server>
```

Required/Optional

Required. You can have a maximum of one `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.

The `server` element attribute is described in Table 7–25.

Table 7–25 Attribute of the `server` Element

Attribute	Valid Values	Description
`version`	string	the version of the Reports Server

7.2.2 Dynamic Environment Switching

Dynamic environment switching enables you to dynamically change the environment after Reports Server is started, or for a specific job request. This means that one instance of Reports Server can serve reports with any arbitrary environment settings, such as language, currency, and display settings.

To enable dynamic environment switching, you must 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 of the following ways:

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

to Section 7.2.1.8, "engine".

- Set the value of the ENVID command line keyword to the id of the environment element, to make the environment settings effective for the current report job request. For more information, refer to Section A.6.5, "ENVID".

7.2.2.1 Examples

The following examples illustrate the use of dynamic environment switching.

Example 1

Suppose that you want 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 (NLS_CURRENCY) would be set to Yen (¥), the currency of Japan.
- If Reports Server is running on UNIX, then DISPLAY must 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 in either of the following ways:

- Use the defaultEnvId attribute of the engine element in the Reports Server configuration file as follows:

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

  The value JP identifies the environment element in the Reports Server configuration file. The initial engines will be spawned with the environment settings specified in this environment element.

- Set the ENVID command line keyword, as follows:

  ```
  http://machine_name:port/reports/rwservlet?SERVER=server_name
  &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 keyword and matches the specified id (in this case, JP) to the corresponding id of the environment element 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 maxEngine 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.

  If you do not pass ENVID 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.

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://machine_name:port/reports/rwservlet?SERVER=server_name
&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; that is, without Oracle Reports Servlet (rwservlet).

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 key map file. See Section 18.14, "Using a Key Map File".

7.2.2.2 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 Reports 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 Oracle Reports, 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
  - Go to the WebLogic Administration Console, navigate to the **Server Start** tab and specify the `oracle.home` and `oracle.instance` parameters.
  - 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 Oracle Reports, 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_INSTANCE/config/reports/bin/reports.sh`), 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 switching 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 `rwserver.conf` file.
3. Copy the environment variable value in the `reports.sh` file to the `rwserver.conf` file. For example:

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

Note: In the above example, the Samples should be downloaded from OTN location http://www.oracle.com/technetwork/middleware/reports/downloads/index.html and then be copied at local location `C:\Samples`.

---

4. Add the `defaultEnvId` value to the appropriate tag in the `rwserver.conf` file.For example, add the `defaultEnvId` attribute to the `engine` element 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="90000" defaultEnvId="default">
```

5. Now run the report.

7.3 Oracle Reports Servlet Configuration File

The configuration settings for the Oracle Reports Servlet (`rwservlet`) component of Oracle Reports Services are stored in the XML file `rwservlet.properties`, located in the directory specified in Table 7–1.

For Windows, note that `rwservlet.properties` uses double backslashes (`\\`) instead 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 `rwservlet.properties` file, the path:

```
d:\DomainHome\config\fmwconfig\components\ReportsServerComponent\ServerName\filename.ext
```

becomes:

```
d:\\DomainHome\\config\\fmwconfig\\components\\ReportsServerComponent\\ServerName\\filename.ext
```

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

```
DomainHome/config/fmwconfig/components/ReportsServerComponent/ServerName/filename.
ext
```

7.3.1 Oracle Reports Servlet Configuration Elements

The `rwservlet.xsd` file provides the following data type definitions for configuring `rwservlet.properties` elements:

- rwservlet
  - server
  - singlesignon
  - inprocess
  - reports_servermap
  - cookie
  - defaultcharset
  - webcommandaccess
  - allowhtmltags
  - helpurl
  - imageurl
  - reloadkeymap
  - dbauth
  - sysauth
  - errortemplate
  - diagtags
  - cluster
  - oidconnection
  - allowauthid
  - enabledbproxy

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

Note that these are XML elements, and XML is case-sensitive. Additionally, when you add any of these elements to the `rwservlet.properties` configuration file, you **must follow the order of elements** as described in `rwservlet.xsd`.

7.3.1.1 rwservlet

The `rwservlet` element is defined in `rwservlet.xsd` as follows:

```
<xs:element name="rwservlet">
  <xs:complexType>
   <xs:sequence>
      <xs:element name="server" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="singlesignon" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="inprocess" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="reports_servermap" type="xs:string" minOccurs="0"
maxOccurs="1"/>
```

```
      <xs:element ref="cookie" minOccurs="0" maxOccurs="1"/>
      <xs:element name="defaultcharset" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      <xs:element ref="webcommandaccess" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="allowhtmltags" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="helpurl" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="imageurl" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="reloadkeymap" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="dbauth" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="sysauth" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="errortemplate" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      <xs:element ref="diagtags" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="cluster" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="oidconnection" minOccurs="0" maxOccurs="1"/>
      <xs:element name="allowauthid" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="enabledbproxy" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example

In `rwservlet.properties`, the `rwservlet` element may be specified as shown in this example:

```
<rwservlet>
  one or more element specifications
</rwservlet>
```

Required/Optional

Required. You can have a maximum of one `rwservlet` element in a given configuration file.

Description

The `rwservlet` element opens and closes the content area of the Oracle Reports Servlet (`rwservlet`) configuration file. In terms of the file's hierarchy, all the other elements are subordinate to the `rwservlet` element.

The `rwservlet` element includes the following sub-elements in its definition:

- server
- singlesignon
- inprocess
- reports_servermap
- cookie
- defaultcharset
- webcommandaccess
- allowhtmltags
- helpurl

- imageurl
- reloadkeymap
- dbauth
- sysauth
- errortemplate
- diagtags
- cluster
- oidconnection
- enabledbproxy

7.3.1.1.1 server

The `server` sub-element of `rwservlet` is defined in `rwservlet.xsd` as follows:

```
<xs:element name="server" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In `rwservlet.properties`, the `server` element may be specified as shown in this example:

```
<server>myserver</server>
```

Description

The `server` element specifies the name of the in-process server. If a Reports Server name is not specified, for example, in the runtime URL, `rwservlet` starts the in-process server (if not started already) with the name specified by the `server` element, and submits the job to it.

If the `server` element is not specified, the default in-process server name is: `rep_hostname`.

When the inprocess element specifies `no`, `rwservlet` tries to bind to an external server with the name specified by the `server` element.

7.3.1.1.2 singlesignon

The `singlesignon` sub-element of `rwservlet` is defined in `rwservlet.xsd` as follows:

```
<xs:element name="singlesignon" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In `rwservlet.properties`, the `singlesignon` element may be specified as shown in this example:

```
<singlesignon>yes</singlesignon>
```

Description

The `singlesignon` element specifies whether or not OracleAS Single Sign-On is enabled:

- `yes` (default): OracleAS Single Sign-On is enabled.
- `no`: OracleAS Single Sign-On is not enabled.

For more information about OracleAS Single Sign-On, refer to Chapter 17, "Configuring and Administering Oracle Single Sign-On".

7.3.1.1.3 inprocess

The inprocess sub-element of rwservlet is defined in rwservlet.xsd as follows:

```
<xs:element name="inprocess" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In rwservlet.properties, the inprocess element may be specified as shown in this example:

```
<inprocess>yes</inprocess>
```

Description

The inprocess element specifies whether or not to run Reports Server within the same process as Oracle Reports Servlet (rwservlet):

- yes (default): Reports Server run within the same process as Oracle Reports Servlet (rwservlet).
- no: Reports Server does not within the same process as Oracle Reports Servlet (rwservlet).

> **Note:** The pros and cons of running an in-process server are explored in Chapter 2, "Understanding the Oracle Reports Services Architecture".

For troubleshooting printing and font issues when using the in-process server, see Section D.1.10, "Printing and Font Errors When Using In-process Reports Server".

7.3.1.1.4 reports_servermap

The reports_servermap sub-element of rwservlet is defined in rwservlet.xsd as follows:

```
<xs:element name="reports_servermap" type="xs:string" minOccurs="0"
            maxOccurs="1"/>
```

Example

In rwservlet.properties, the reports_servermap element may be specified as shown in this example:

```
<reports_servermap>
  dev_cluster:dev_server;prd_cluster:prd_server;qa_cluster:qa_server
</reports_servermap>
```

Description

In Oracle Reports 11*g* Release 1 (11.1.1), Reports Server clustering was deprecated. An Oracle Forms Services application from prior releases that includes a Reports Server cluster name will fail to bind to the Reports Server cluster it references.

To resolve this issue, the reports_servermap element maps a cluster name to a Reports Server name. This avoids the necessity to change the cluster name in all Oracle Forms Services applications.

An Oracle Forms Services application can call Oracle Reports in the following ways:

- Using RUN_REPORT_OBJECT. If the call specifies a Reports Server cluster name instead of a Reports Server name, the REPORTS_SERVERMAP environment variable must be set in the Oracle Forms Services default.env file.

If your Oracle Forms Services application uses multiple Reports Server cluster names, you can map each of those cluster names to a different Reports Server using REPORTS_SERVERMAP, as follows:

```
REPORTS_SERVERMAP=cluster1:repserver1;cluster2:repserver2;cluster3:repserver3
```

For example, if your Oracle Forms Services application includes 3 clusters with names dev_cluster, prd_cluster, and qa_cluster in 10*g* Release 1 (9.0.4), you can map these cluster names to respective server names in later releases, as follows:

```
REPORTS_SERVERMAP=dev_cluster:dev_server;prd_cluster:prd_server;qa_cluster:qa_
server
```

See the *Forms Services Deployment Guide*.

- Using WEB.SHOW_DOCUMENT. In this case, the request is submitted to rwservlet. If the call specifies a Reports Server cluster name instead of a Reports Server name, the reports_servermap element must be set in the rwservlet.properties file. For example:

```
<reports_servermap>
  cluster:repserver
</reports_servermap>
```

where

cluster is the Reports Server cluster name that was present in prior releases (Oracle Reports 9*i* and 10*g* Release 1 (9.0.4)).

repserver is the Reports Server name in later releases.

When reports_servermap is set in rwservlet.properties, any request to *cluster* in the Oracle Forms Services application is redirected to *repserver*.

7.3.1.1.5 cookie

The cookie element is defined in rwservlet.xsd as follows:

```
<xs:element name="cookie">
  <xs:complexType>
    <xs:attribute name="cookieexpire" use="required" type="xs:integer"/>
    <xs:attribute name="encryptionkey" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example

In rwservlet.properties, the cookie element may be specified as shown in this example:

```
<cookie cookieexpire="30" encryptionkey="reports"/>
```

Required/Optional

Optional.

Description

The cookie element specifies an expiration time and encryption key for cookies, which 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 difference 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.

The `cookie` element attributes are described in Table 7–26.

Table 7–26 Attributes of the `cookie` Element

Attribute	Valid Values	Description
`cookieexpire`	Integer	**Default:** `30` The lifetime (in minutes) of the database and system authentication cookie.
`encryptionkey`	any character string	The encryption key to be used to encrypt the user name and password of the database and system authentication cookies.

7.3.1.1.6 defaultcharset

The `defaultcharset` sub-element of `rwservlet` is defined in `rwservlet.xsd` as follows:

```
<xs:element name="defaultcharset" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In `rwservlet.properties`, the `defaultcharset` property may be specified as shown in this example:

```
<defaultcharset>JA16EUC</defaultcharset>
```

Description

The `defaultcharset` element specifies the character encoding for decoding non-ASCII escaped characters in the request URL or non-ASCII characters in the Parameter Form input. This ensures that `rwservlet` uses the required encoding when decoding the parameter name and value.

You can set the `defaultcharset` element to either:

- The database's `NLS_CHARACTERSET` (for example, `JA16EUC`).
- The IANA-defined character set (for example, `EUC-JP`).

> **Note:** To use non-ASCII characters in user parameter names and values when using the Event-Driven Publishing API, you must ensure that the `defaultcharset` element in the `rwservlet.properties` file matches the value of the `DEFAULTCHARSET` parameter in your parameter list. See Section 21.1.3, "Including non-ASCII Characters in Parameter Names and Values".

7.3.1.1.7 webcommandaccess

The `webcommandaccess` element is defined in `rwservlet.xsd` as follows:

```
<xs:element name="webcommandaccess">
   <xs:simpleType>
    <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="L0"/>
        <xs:enumeration value="L1"/>
        <xs:enumeration value="L2"/>
        <xs:enumeration value="NO"/>
```

```
        <xs:enumeration value="YES"/>
      </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Example

In `rwservlet.properties`, the `webcommandaccess` property may be specified as shown in this example:

```
<webcommandaccess>L1</webcommandaccess>
```

Description

The `webcommandaccess` element specifies access permission for `rwservlet` keywords (Web commands) for a non-secure server.

> **Note:** For secure Reports Server, Reports Server verifies the user's privileges based on the entries in Oracle Internet Directory.

Valid settings are:

- `L0`: no Web commands allowed.
- `L1`: only end user Web commands allowed (GETJOBID, KILLJOBID, SHOWAUTH, SHOWJOBID).
- `L2`: administrator Web commands (DELAUTH, GETSERVERINFO, KILLENGINE, PARSEQUERY, SHOWENV, SHOWJOBS, SHOWMAP, SHOWMYJOBS) are also allowed. AUTHID is required to run administrator commands.
- `NO`: for backward compatibility with `DIAGNOSTIC=NO` in 10*g* (`rwservlet.properties`).
- `YES`: for backward compatibility with `DIAGNOSTIC=YES` in 10*g* (`rwservlet.properties`).

For L2 Web command access, you do not need to pass the `authid`. The authid parameter is required only for the STOPSERVER command irrespective of the webcommandaccess value.

7.3.1.1.8 allowhtmltags

The `allowhtmltags` sub-element of `rwservlet` is defined in `rwservlet.xsd` as follows:

```
<xs:element name="allowhtmltags" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In `rwservlet.properties`, the `allowhtmltags` element may be specified as shown in this example:

```
<allowhtmltags>yes</allowhtmltags>
```

Description

The `allowhtmltags` element specifies whether or not to allow HTML code to be entered in the URL when running a report:

- `no` (default): HTML code in the URL is disallowed.
- `yes`: HTML code in the URL is allowed.

Note: Any HTML code included as part of a report request URL might lead to a security compromise as it causes certain browsers to execute any script or code in the URL.

7.3.1.1.9 helpurl

The `helpurl` sub-element of `rwservlet` is defined in `rwservlet.xsd` as follows:

```
<xs:element name="helpurl" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In `rwservlet.properties`, the `helpurl` element may be specified as shown in this example:

```
<helpurl>http://myserver/help_file/help_topic.htm</helpurl>
```

Description

The `helpurl` element specifies the name of a help file to be used instead of the default (*ORACLE_HOME*`\reports\templates\help.htm`).

The `rwservlet HELP` keyword (Web command) displays either the default help file, or the help file specified by the `helpurl` element.

Note: For more about the `HELP` keyword, see Section A.6.12, "HELP".

7.3.1.1.10 imageurl

The `imageurl` sub-element of `rwservlet` is defined in `rwservlet.xsd` as follows:

```
<xs:element name="imageurl" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In `rwservlet.properties`, the `imageurl` property may be specified as shown in this example:

```
<imageurl>http://machine_name:port/reports/rwservlet</imageurl>
```

Description

The `imageurl` element specifies the location of reports' dynamically generated images.

This element applies to JSPs that do not run through Oracle Reports Servlet (`rwservlet`). 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 `rwservlet` automatically have this protection.

7.3.1.1.11 reloadkeymap

The `reloadkeymap` sub-element of `rwservlet` is defined in `rwservlet.xsd` as follows:

```
<xs:element name="reloadkeymap" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In `rwservlet.properties`, the `reloadkeymap` element may be specified as shown in this example:

```
<reloadkeymap>yes</reloadkeymap>
```

Description

The reloadkeymap element specifies whether the key map file (cgicmd.dat) should be reloaded each time rwservlet receives a request:

- no (default): Key map file is not reloaded when rwservlet receives a request.
- yes: Key map file is reloaded when rwservlet receives a request.

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 element specifies no in a production environment and yes in a testing environment.

7.3.1.1.12 dbauth

dbauth is the html template used to enter the database information. If the user does not enter the database information while giving reports request, the reports servlet challenges the user to enter the db info in the HTML template.

The dbauth sub-element of rwservlet is defined in rwservlet.xsd as follows:

```
<xs:element name="dbauth" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In rwservlet.properties, the dbauth element may be specified as shown in this example:

```
<dbauth>rwdbauth.htm</dbauth>
```

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

ORACLE_HOME\reports\templates

Description

The dbauth element specifies the location and filename of the HTML templates, if you wish to customize the login dialog boxes with your company logo, linked buttons, or any other HTML you care to use. By default, the file name is rwdbauth.htm.

7.3.1.1.13 sysauth

sysauth is the HTML template used to enter the authentication information.

The sysauth sub-element of rwservlet is defined in rwservlet.xsd as follows:

```
<xs:element name="sysauth" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In rwservlet.properties, the sysauth element may be specified as shown in this example:

```
<sysauth>rwsysauth.htm</sysauth>
```

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

ORACLE_HOME\reports\templates

Description

The sysauth element specifies the location and filename of the HTML templates, if you wish to customize login dialog boxes for a secure report with your company logo,

linked buttons, or any other HTML you care to use. By default, the file name is rwsysauth.htm.

7.3.1.1.14 errortemplate

The errortemplate sub-element of rwservlet is defined in rwservlet.xsd as follows:

```
<xs:element name="errortemplate" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Example

In rwservlet.properties, the errortemplate element may be specified as shown in this example:

```
<errortemplate>rwerror.htm</errortemplate>
```

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

```
ORACLE_HOME\reports\templates
```

Description

The errortemplate element specifies the name and location of your error message template. By default, the file name is rwerror.htm.

The error message 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 using the errortemplate element to specify your own HTML framework for automatically generated error messages.

The character set of the default error message template (rwerror.htm) is iso-8859-1 to ensure consistency across all platforms.

7.3.1.1.15 diagtags

The diagtags element is defined in rwservlet.xsd as follows:

```
<xs:element name="diagtags">
  <xs:complexType>
    <xs:attribute name="diagbodytags" use="required" type="xs:string"/>
    <xs:attribute name="diagheadtags" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Required/Optional

Optional.

Description

The diagtags element specifies additional HTML encoding in the <body> and <head> tags in the output files associated with diagnostic and debugging output. You can use these to include formatting options to make diagnostic and debugging output easier to read.

The diagtags element attributes are described in Table 7–27.

Table 7–27 Attributes of the `diagtags` Element

Attribute	Valid Values	Description
`diagbodytags`	HTML code for `<body>` tag	HTML code to add between the `<body>` and `</body>` tags.
`diagheadtags`	HTML code for `<head>` tag	HTML code to add between the `<head>` and `</head>` tags.

7.3.1.1.16 cluster

Note: For information about Oracle Fusion Middleware-level techniques for high availability, refer to Section 2.4, "Setting Up a High Availability Environment".

7.3.1.1.17 oidconnection

The `oidconnection` element is defined in `rwservlet.xsd` as follows:

```
<xs:element name="oidconnection">
  <xs:complexType>
    <xs:attribute name="oidcon_appentity" use="required" type="xs:string"/>
    <xs:attribute name="oidcon_passwdkey" use="required" type="xs:string"/>
    <xs:attribute name="oidcon_url" use="required" type="xs:string"/>
    <xs:attribute name="oidcon_init" type="xs:integer"/>
    <xs:attribute name="oidcon_increment" type="xs:integer"/>
    <xs:attribute name="oidcon_timeout">
        <xs:simpleType>
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="1"/>
    </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

In `rwservlet.properties`, the `oidconnection` element may be specified as shown in this example:

```
<oidconnection>
  oid_appentity="reportsapp.idc.oracle.com>
  oidcon_init="10"
  oidcon_increment="10"
  oidcon_timeout="1">
</oidconnection>
```

Description

The `oidconnection` element specifies Oracle Internet Directory for `rwservlet`.

For Reports Server, you can specify Oracle Internet Directory connection pooling parameters using the `oidconnection` element in the server configuration file, as described in Section 7.2.1.15, "oidconnection".

The `oidconnection` element attributes are described in Table 7–28.

Table 7–28 Attributes of the `oidconnection` Element

Attribute	Valid Values	Description
`oidcon_appentity`	N/A	**Default:** Set at install time Oracle Internet Directory App entity created at install time for internal use of Reports.
`oidcon_passwdkey`	N/A	**Default**: Set at install time Random password key created to connect to Oracle Internet Directory for internal use of Reports.
`oidcon_url`	N/A	**Default**: Set at install time Oracle Internet Directory Url to connect to oid
`oidcon_init`	number	**Default**: `10` Initial number of Oracle Internet Directory connections to be created when `rwservlet` is initialized.
`oidcon_increment`	number	**Default**: `10` Number of connections to be incremented when all connections are used up.
`oidcon_timeout`	number	**Default**: `0` (no timeout) Time in seconds for which a connection can be idle.

7.3.1.1.18 allowauthid

The `allowauthid` sub-element of rwservlet is defined in rwservlet.xsd as follows:

```
<xs:element name="allowauthid" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Description

`allowauthid` is the element to be added in rwservlet.properties either to enable or disable passing of the authid through an URL.

Example

In `rwservlet.properties`, the `allowauthid` element may be specified as shown in this example:

```
<allowauthid> yes </allowauthid>
```

Required/Optional

Optional.

Default

By default the `allowauthid` is set to `Yes`. If the authid is set to `No`, authorization through an URL is disabled and Single Sign-On should be used to enter the username and password.

7.3.1.1.19 enabledbproxy

The `enabledbproxy` sub-element of rwservlet is defined in rwservlet.xsd as follows:

```
<xs:element name="enabledbproxy" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

Description

enabledbproxy is the element to be added in the rwservlet.properties file to make the dbproxy feature work through the rwservlet.

Example

In rwservlet.properties, the enabledbproxy element may be specified as shown in this example:

```
<enabledbproxy> yes </enabledbproxy>
```

Required/Optional

Optional.

Default

By default, the enabledbproxy is set to Yes.

7.3.2 Specifying an Alternate Oracle Reports Servlet Configuration File

Perform the following steps to specify an alternate Oracle Reports Servlet Configuration File:

1. Log in to the **WebLogic Server Administration Console**.
2. Under Domain Structure in the left pane, click **Environment**. The **Summary of Environment** page is displayed.
3. In this page, click **Servers**. The **Summary of Servers** page is displayed.
4. From the list of servers, click **WLS_REPORTS**. The **Settings for WLS_ REPORTS** page is displayed.
5. Click the **Server Start** tab.
6. Add the following entry in the **Arguments** field:

   ```
   -DServletPropFile=your_servlet_properties_file
   ```

7. Click **Save.**
8. Restart **WLS_REPORTS**.

By default, Oracle Reports Servlet (rwservlet) uses the rwservlet.properties file as the configuration file. If you are running multiple Oracle WebLogic Server instances with reports installed on the same Oracle Fusion Middleware and wish to use different configuration files, you can do so by adding the following parameter in the WLS_ REPORTS startup parameter section in the WebLogic Server Administration Console:

```
-DServletPropFile=your_servlet_properties_file
```

7.4 Oracle Reports Bridge Configuration File

The Oracle Reports Bridge configuration settings for the Reports Server component of Oracle Reports Services are stored in the XML file rwbridge.conf.

The bridgeconfig.xsd file contains data type definitions for rwbridge.conf elements and attributes. See Section 7.4.1, "Oracle Reports Bridge Configuration Elements".

These files are located in the directories specified in Table 7–1.

7.4.1 Oracle Reports Bridge Configuration Elements

The `bridgeconfig.xsd` contains the data type definitions for the various Oracle Reports Bridge configuration file (rwbridge.conf) elements and attributes.

The Oracle Reports Bridge acts as a gateway for packets that are broadcast by Reports Server and Reports Client across Farms. For example, in a sample setup, Oracle Reports components are installed on different Farms: Oracle Reports Servlet is in Farm A and Reports Server is in Farm B. To achieve this configuration, the Oracle Reports Bridge has to be started on each Farm. Bridge configuration will include the host and port settings. The Oracle Reports Bridge in Farm A will contact the Oracle Reports Bridge in Farm B through reliable TCP to retrieve the server information on Farm B, and vice versa. See Section 2.2.4.1.2, "Server Discovery Across Subnets".

To start and stop the Oracle Reports Bridge, refer to Chapter 5, "Starting and Stopping Oracle Reports Services".

Oracle Reports creates the configuration file rwbridge.conf when the Oracle Reports Bridge is started for the first time. This file is generated based on the settings in the `rwbridge.template` file.

In the configuration file, rwbridge.conf, modify the `identifier` element to specify the *`username/password`* and set the `encrypted` attribute to `no`. This indicates that the password is not encrypted. This password will be encrypted once the Oracle Reports Bridge is started.

For example:

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

Usage Notes

- If the `identifier` element is commented, then it is possible to stop the Oracle Reports Bridge without specifying `authid`.
- It is not possible to stop the Oracle Reports Bridge remotely.

See Also: Section 7.4.1.2, "identifier"

The `bridgeconf.xsd` file provides the following data type definitions for configuring `rwbridge.conf` elements and attributes:

- bridge
- identifier
- remoteBridge
- remoteBridges

7.4.1.1 bridge

The `bridge` element is defined in `bridgeconfig.xsd` as follows:

```
<xs:element name="bridge">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="identifier" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="remoteBridges" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>

    <xs:attribute name="version" use="required" type="xs:string"/>

    <xs:attribute name="port" default="14011">
```

```
      <xs:simpleType>
      <xs:restriction base="xs:integer">
    <xs:minInclusive value="14001"/>
      </xs:restriction>
      </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="timeout" use="required">
<xs:simpleType>
<xs:restriction base="xs:integer">
<xs:minInclusive value="100"/>
</xs:restriction>
</xs:simpleType>
    </xs:attribute>

    </xs:complexType>
</xs:element>
```

Example

```
<bridge version="11.1.1.1.0" port="14011" timeout="12000">
   <!--networkConfig file="rwnetwork.conf" ></networkConfig-->
   <!--identifier encrypted="no"
       >%USERNAME%/%PASSWORD%</identifier-->
   <!--trace traceOpts="trace_all"></trace-->
   <!-- Specify one or more remote bridges inside remoteBridges element -->
   <!--remoteBridges>
      <remoteBridge host="%HOST%" port="%PORT%"></remoteBridge>
   </remoteBridges-->
</bridge>
```

Required/Optional

Required. You can have a maximum of one open tag and one close tag in the `bridge` element in a given configuration file.

Description

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

The `bridge` element attributes are described in Table 7–29.

Table 7–29 Attributes of the `bridge` Element

Attributes	Valid Values	Description
`version`	`11.1.1.1.0`	The bridge version.
`port`	The allotted range for Oracle Reports Bridge component; that is, `14011` to `14020`.	The port on which the bridge will listen.
`timeout`	`1000`	Value in milliseconds (ms). The bridge will wait for this period for a response from a remote bridge.

Note: The default port value for the bridge configuration file is assigned when you install Oracle Fusion Middleware. The `rwbridge.template` file contains this default port, which is used to generate the configuration file for the bridge. The configuration file name for a bridge is `ORACLE_INSTANCE/config/ReportsBridgeComponent/<bridge name>/rwbridge.conf`.

If you want to customize the port number for the bridge, you must specify a valid port range reserved for the Oracle Reports Bridge bridge (`14011` to `14020`).

7.4.1.2 identifier

The `identifier` element is defined in `bridgeconfig.xsd` as follows:

```
<xs:element name="identifier">
  <xs:complexType mixed="true">
      <xs:attribute name="encrypted" default="no">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="yes"/>
          <xs:enumeration value="no"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

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

Required / Optional

Optional. If this element is commented, then the Oracle Reports Bridge will not perform a security check when the bridge shutdown command is issued.

Description

The `identifier` element ensures that the Oracle Reports Bridge performs a security check before shutting down.

To set the value of the `identifier` element:

1. Uncomment the `identifier` element in the bridge configuration file.
2. Set the value to the administrator username/password, set the attribute `encrypted=no`, so that the username/password will be encrypted when the Oracle Reports Bridge is restarted.

   For example:

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

3. Start the Oracle Reports Bridge.

Once this element is set, only the administrator will be able to shut down the bridge by specifying the `username/password` in the command line.

See Also: Section 5.2.1, "Starting, Stopping, and Restarting the Oracle Reports Bridge from the Oracle Process Manager and Notification Server"

7.4.1.3 remoteBridge

The `remoteBridge` element is defined in `bridgeconfig.xsd` as follows:

```
<xs:element name="remoteBridge">
  <xs:complexType>
    <xs:attribute name="host" use="required" type="xs:string"/
    <xs:attribute name="port" use="required">
       <xs:simpleType>
    <xs:restriction base="xs:integer">
    <xs:minInclusive value="14001"/>
    </xs:restriction>
    </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

```
<remoteBridge host="myhost.mydomain.com" port="14022"></remoteBridge>
```

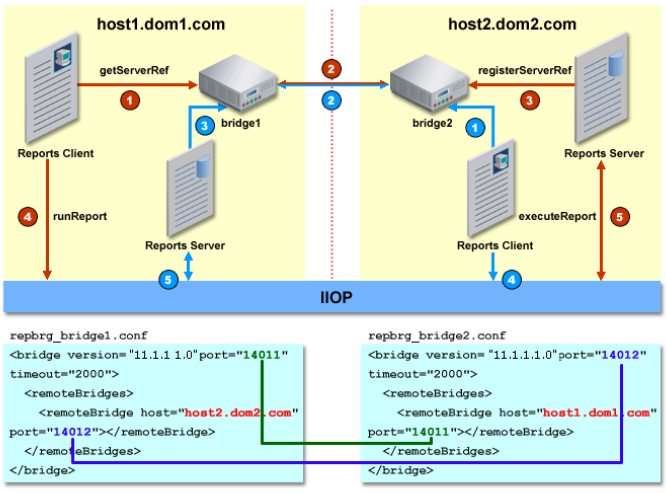
Required/Optional

Optional. You can have one or more `remoteBridge` elements in your bridge configuration file.

Description

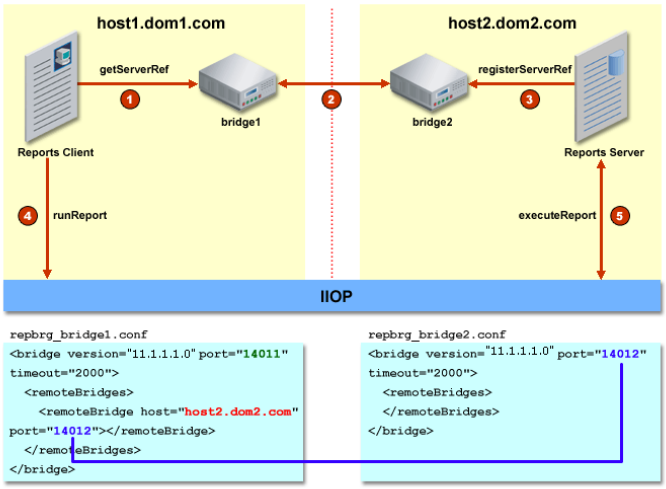
The `remoteBridge` element specifies the host and port on which remote bridges are running.

Figure 7–1 Oracle Reports Bridge Configuration (Two-Way)



If you specify the optional `remoteBridge` element(s) in the `repbrg_`*`bridgename`*`.conf`, then the bridge will act as a two-way bridge. That is, the bridge can get server references from remote bridges.

Figure 7–2 Oracle Reports Bridge Configuration (One-Way)



If you do *not* specify the optional `remoteBridge` element(s) in the `repbrg_bridgename.conf`, then the bridge will act as a one-way bridge. That is, the bridge can only serve remote bridges. It cannot connect to remote bridges to get the server reference.

The `remoteBridge` element attributes are described in Table 7–30.

Table 7–30 Attributes of the `remoteBridge` Element

Attributes	Valid Values	Description
`host`	Host name/IP address of the remote bridge.	The host name or the IP address of the machine where the remote bridge is running.
`port`	Port number of the remote bridge.	The port number of the remote bridge element.

7.4.1.4 remoteBridges

The `remoteBridges` element is defined in `bridgeconfig.xsd` as follows:

```
<xs:element name="remoteBridges">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="remoteBridge" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example

```
<remoteBridges>
  <remoteBridge host="myhost.mydomain.com" port="14022"></remoteBridge>
</remoteBridges>
```

Required/Optional

Optional. If this entry is not specified, then this bridge will not contact any remote bridge to get a Reports Server reference. However, remote bridges can contact this bridge to get the references of Reports Servers running in this farm.

Description

The `remoteBridges` element can contain zero or more remoteBridge elements.

7.5 Network Configuration File

The network configuration settings for the Oracle Reports Bridge component of Oracle Reports Services are stored in the XML file `rwnetwork.conf`.

The `rwnetworkconf.xsd` contains data type definitions for `rwnetwork.conf` elements and attributes. See Section 7.5.1, "Network Configuration Elements".

These files are located in the directories specified in Table 7–1.

7.5.1 Network Configuration Elements

The `rwnetworkconf.xsd` file provides the following data type definitions for configuring `rwnetwork.conf` elements and attributes:

- discoveryService
- multicast
- namingService

7.5.1.1 discoveryService

The `discoveryService` element is defined in `rwnetwork.xsd` as follows:

```
<xs:element name="discoveryService">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="multicast"/>
      <xs:element ref="namingService"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Required/Optional

Required. You can have a maximum of one open tag and one close tag in the `discoveryService` element in a given configuration file.

Description

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

The `discoveryService` element has no attributes. It includes two sub-elements:

- `multicast`: see Section 7.5.1.2, "multicast"
- `namingService`: see Section 7.5.1.3, "namingService"

7.5.1.2 multicast

The `multicast` element is defined in `rwnetwork.xsd` as follows:

```
<xs:element name="multicast">
  <xs:complexType>
    <xs:attribute name="channel" default="228.5.6.7" type="xs:string"/>
    <xs:attribute name="port"  default="14021">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="timeout" use="required">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="100"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="retry"  default="3">
         <xs:simpleType>
         <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
         </xs:restriction>
         </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

```
<multicast channel="228.5.6.7" port="one of the port in alotted AS ports"
           timeout="1000" retry="3"/>
```

Required/Optional

Conditional. The `namingService` and `multicast` elements are mutually exclusive; that is, only one of these elements can be configured at a time.

Description

The `multicast` element contains the necessary information to identify where Reports Server is running the built-in broadcast mechanism. By default, `multicast` is specified in `rwnetwork.conf`.

The `multicast` element attributes are described in Table 7–31.

Table 7–31 Attributes of the `multicast` Element

Attributes	Valid Values	Description
`channel`	Broadcast channel	The broadcast channel used by the Reports Server.
`port`	Broadcast port	The broadcast port used by the Reports Server.
`timeout`	Time (in milliseconds) it should wait for response. The optimum value for this setting is `1000`.	The Reports Client will wait for the specified timeout period for a response from the Reports Server.

Table 7–31 (Cont.) Attributes of the `multicast` Element

Attributes	Valid Values	Description
`retry`	Retry count	The Reports Client will retry for the specified number of times, if there is no response from the Reports Server after the timeout period.

Note: It is strongly recommended that you do not change the default channel and port unless it is absolutely necessary. The default port value for `rwnetwork.conf` is assigned when you install Oracle Fusion Middleware.

If you want to customize `rwnetwork.conf`, you must specify a valid port range reserved for Reports Server (14021 to 14030). If you are using the Oracle Reports Bridge for discovering Reports Servers across subnets, you should set the `timeout` and `retry` values carefully for the bridge to function correctly. Refer to Table 7–31 for setting the timeout value.

7.5.1.3 namingService

The `namingService` element is defined in `rwnetwork.xsd` as follows:

```
<xs:element name="namingService">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute name="host"  use="required" type="xs:string"/>
    <xs:attribute name="port" use="required">
        <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Example

```
<namingService name="Cos" host="mymachine.mydomain.com" port="14021"/>
```

Required/Optional

Conditional. The `namingService` and `multicast` elements are mutually exclusive; that is, only one of these elements can be configured at a time.

Description

The `namingService` element contains the necessary information required to be able to identify the host name and the port where the COS naming service is running. `namingService` is not supported on Reports High Availability. Specify this element only when the built-in broadcast mechanism is not suitable for your environment, as in the following scenarios:

- Oracle Reports is installed on a machine that is connected to a network using VPN.

- You want to avoid broadcast traffic on your network.

See Section 2.2.4.2, "Server Discovery Using the COS Naming Service".

The `namingService` element attributes are described in Table 7–32.

Table 7–32 Attributes of the `namingService` Element

Attributes	Valid Values	Description
`name`	`Cos`	The descriptive name of the naming service.
`host`	Host name/IP	The host name of the machine where the naming service is running.
`port`	Port number	The port number of the machine where the naming service is running.

7.6 Configuring the URL Engine

Reports Server includes a URL engine that can take the contents of any URL and distribute them. The URL engine enables 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, Oracle 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="rwurl" 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 preceding 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 18, "Running Report Requests".

7.7 Entering Proxy Information

Some features of Oracle 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 Oracle Reports, proxy information is stored in the Reports Server configuration file (`rwserver.conf`). You can specify proxy information in either of the following ways:

- Editing the Server Configuration File

7.7.1 Editing the Server Configuration File

To specify proxy information by editing the server configuration file (`rwserver.conf`) directly, add the `proxyServer` element, as described in Section 7.2.1.23, "proxyServer".

7.8 Configuring Reports Server with the Node Manager Reports Process Management

The best way to start, shut down, 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 Fusion Middleware components, including Reports Server. Out-of-the-box, Oracle Reports components are managed with OPMN for death detection and recovery, providing an enhanced health check mechanism in Oracle Reports 12*c* Release (12.2.1.3).

If the process fails for any reason, Nodemanager restarts Reports Server for you automatically.

During installation of Oracle Fusion Middleware, Reports Servers are automatically configured in Node Manager.

7.8.1 Starting and Stopping Components

Start / Stop of components is done using WLST commands. The reports process management plugin running in Node Manager does the actual process management of reports server / bridge. If reports processes die, Node Manager detects it and restarts the processes.

In these examples:

- Name of reports server component - `repsvr1`

- Name of reports bridge component - `repbrd1`
- Domain Home - `/scratch/rrpai/wls2/user_projects/domains/test1`

7.8.1.1 FMW Script

Use the following commands:

```
$DOMAIN_HOME/bin/startComponent.sh  repsvr1
$DOMAIN_HOME/bin/stopComponent.sh  repsvr1

$DOMAIN_HOME/bin/startComponent.sh  repbrd1
$DOMAIN_HOME/bin/stopComponent.sh  repbrd1
```

7.8.1.2 WLST Start / Stop

WLST Online

```
setenv ORACLE_HOME  /scratch/rrpai/wls2

$ORACLE_HOME/oracle_common/common/bin/wlst.sh
connect("weblogic","welcome1", "localhost:7001")
domainRuntime()
# reports server
cd('/SystemComponentLifeCycleRuntimes/repsvr1')
# start reports server
cmo.start(java.util.Properties())
# get state of reports server
cmo.getState()
# stop reports server
cmo.shutdown(java.util.Properties())

# reports bridge
cd('/SystemComponentLifeCycleRuntimes/repbrd1')
# start reports bridge
cmo.start(java.util.Properties())
# get state of reports bridge
cmo.getState()
# stop reports bridge
cmo.shutdown(java.util.Properties())
```

WLST Offline

```
$ORACLE_HOME/oracle_common/common/bin/wlst.sh
# nmConnect('weblogic', 'welcome1', 'localhost', 'NodeManagerPort', 'DomainName',
'DomainHome')
nmConnect('weblogic', 'welcome1', 'localhost', '5556', 'test1',
'/scratch/rrpai/wls2/user_projects/domains/test1')
# reports server
nmStart(serverName='repsvr1', serverType='ReportsServerComponent')
nmKill(serverName='repsvr1', serverType='ReportsServerComponent')

# reports bridge
nmStart(serverName='repbrd1', serverType='ReportsBridgeComponent')
nmKill(serverName='repbrd1', serverType='ReportsBridgeComponent')
```

7.8.1.3 Files Associated with process control

repsvr1 is the name of reports server

ReportsBridgeComponent is the reports bridge location

The startup properties are managed by nodemanager

```
$DOMAIN_HOME/system_
components/ReportsServerComponent/repsvr1/data/nodemanager/startup.propert
ies
```

Some nodemanager properties are mentioned below:

Table 7–33 nodemanager properties

Property	Description	Value
RestartMax	The number of times Node Manager can attempt to restart the server.	2
RestartInterval	The amount of time Node Manager will spend attempting to restart a failed server. Within this period of time Node Manager will attempt to restart the failed server up to the number defined by RestartMax. By default, Node Manager will attempt to restart a server indefinitely until the `FAILED_NOT_RESTARTABLE` state is reached.	3600
RestartDelaySeconds	The number of seconds Node Manager should wait before attempting to restart the server.	0
AutoRestart	Specifies whether Node Manager can automatically restart this server if it fails.	true

- reports environment and timeout config

  ```
  $DOMAIN_HOME/system_
  components/ReportsServerComponent/repsvr1/data/nodemanager/reports_
  process.xml
  ```

- nodemanager log

  ```
  $DOMAIN_HOME/nodemanager/nodemanager.log
  ```

- stdout logs of reports server, created by NodeManager

  ```
  $DOMAIN_HOME/servers/repsvr1/logs/repsvr1.out
  ```

- nodemanager process control log config

  `$DOMAIN_HOME/nodemanager/nodemanager.properties`

- change log level change this setting

  `LogLevel=INFO`

7.8.1.4 Reports process management configuration file

For example: `$DOMAIN_HOME/system_components/ReportsServerComponent/repsvr1/data/nodemanager/reports_process.xml`

You can configure the following:

- Environment variables to be set for the server
  - If append is set to 'true' it appends the value to current value using platform specific path separator that is, Windows - ';' and Unix - ':'. This is suitable for environment variables like PATH
  - If append is set to 'false' it replaces the environment variable value
- Stop timeout
- This is how nodemanager reports plugin works while stopping reports server
  - first it will attempt to shutdown in 'normal' mode that is, wait for reports jobs running in reports server to finish
  - if in 'timeout' specified in config it does not stop, it will send attempt to shutdown in immediate mode (that is, does not wait for job finish)
  - if in 'timeout' it does not stop again it will force kill process

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<process xmlns="http://xmlns.oracle.com/reports/process"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <environment>
    <variable id="PATH" value="/scratch/oracle/testPath" append="true"/>
    <variable id="MY_PATH" value="/home/oracle/myPath" append="false"/>
    <variable id="MY_ENV" value="my_env_value"/>
  </environment>
  <stop timeout="30"/>
</process>
```

7.8.2 Creating a New Reports Tools Component Type

Use the following commands:

```
$MW_HOME/oracle_common/common/bin/wlst.sh
connect("weblogic","welcome1", "localhost:7001")
help()
help('reports_tools')
help('createReportsToolsInstance')
help('deleteReportsToolsInstance')
createReportsToolsInstance(instanceName='reptools1', machine='M1')
deleteReportsToolsInstance(instanceName='reptools1')
exit()
```

7.8.3 Creating a New Reports Bridge Component Type

Use the following commands:

```
$MW_HOME/oracle_common/common/bin/wlst.sh
connect("weblogic","welcome1", "localhost:7001")
help()
help('reports_bridge')
help('createReportsBridgeInstance')
help('deleteReportsBridgeInstance')
createReportsBridgeInstance(instanceName='repbridge1', machine='M1')
deleteReportsBridgeInstance(instanceName='repbridge1')
exit()
```

7.8.4 Creating a New Reports Server Component Type

Use the following commands:

```
$MW_HOME/oracle_common/common/bin/wlst.sh
connect("weblogic","welcome1", "localhost:7001")
help()
help('reports_server')
help('createReportsServerInstance')
help('deleteReportsServerInstance')
createReportsServerInstance(instanceName='repsvr1', machine='M1')
deleteReportsServerInstance(instanceName='repsvr1')
exit()
```

7.9 Overview of SOA Integration

With 12*c* Release (12.2.1.3), Oracle Reports is integrated with the Oracle Service-Oriented Architecture (SOA) suite, which includes Oracle BPEL Process Manager to automate and monitor reporting requirements.

7.9.1 About BPEL

Business Process Execution Language (BPEL) is the emerging standard for assembling a set of discrete services into an end-to-end process flow, radically reducing the cost and complexity of process integration initiatives. Oracle BPEL Process Manager, a key component of Oracle Fusion Middleware, enables enterprises to orchestrate disparate applications and web services into business processes. The ability to quickly build and deploy these processes in a standards-based manner delivers critical functionality for developing a Service-Oriented Architecture (SOA).

For more information about Oracle BPEL Process Manager, see *Developing SOA Applications with Oracle SOA Suite*.

Users can submit Oracle Reports jobs from the Oracle BPEL Process Manager business process, get the status of report execution, and also invoke an Oracle BPEL Process Manager business process from reports. For example, submit a report request when an order gets approved.

Using Oracle Reports with Oracle BPEL Process Manager involves the following steps:

- Define a business process by using Oracle JDeveloper.
- Call the Oracle Reports web service to start executing the report at an appropriate stage in your business process.
- Submit the reports job synchronously or asynchronously.
- Obtain the status of the report job at any time.

7.10 Configuring Oracle Reports to Communicate with Oracle BPEL Process Manager

Oracle Reports exposes web service named `RWWebService`. For more information about this web service, see Chapter 19, "Using the Oracle Reports Web Service". Oracle Reports web services, which are synchronous in nature, can be used in a BPEL process as a Partner link.

To invoke an Oracle Reports web service asynchronously in a BPEL process, do the following:

- Create an intermediate asynchronous BPEL process that uses the synchronous Oracle Reports web service and provides an interface to the caller for making a callback.
- Deploy this intermediate asynchronous BPEL process as a service in the SOA suite, which can be used as a Partner link.

You can access the WSDL of the Oracle Reports web service at the following location:

http://yourwebserver:port/reports/rwwebservice?

WSDL contains information about the different ways of invoking `RWWebservice`. You can access the XSD file at the following location:

http://yourwebserver:port/reports/rwwebservice?xsd=1

You can save the XSD file on your local machine and use it when creating a BPEL process.

7.10.1 Using RWWebservice to Submit Jobs to the Reports Server

To submit jobs to the Oracle Reports server using RWWebService, perform the following steps:

1. Install Oracle JDeveloper. See *Installing Oracle JDeveloper*.
2. Install Oracle SOA.
3. Start the SOA suite.
4. Create a connection from Oracle JDeveloper to the SOA suite.
5. Create a new application.
6. Create a new asynchronous BPEL process project and use the `RWWebservice.xsd` file to define the input and output parameters of the BPEL process.
7. Create a Partner link that refers to the `RWWebService` WSDL:
   - When prompted by Oracle JDeveloper, click **Yes** to create Partner link types.
   - Select Partner Role.
8. Create a scope in your process.
9. Within the scope, create an Invoke activity and use it to invoke the runJob operation on the `RWWebService` Partner link.
10. Create input and output variables automatically. The input parameter provides input to the `runJob` operation of RWWebService, and the output parameter contains the output of the runJob operation.
11. Within the scope, before the Invoke activity, create an Assign activity:

- In this Assign activity, map the user input parameters to the parameters that `RWWebService` requires.
- Set Param0 = reports job command (string).
- Set Param1 = `true` (boolean, specifying that the job submission must be synchronous).

12. Within the scope, after the Invoke activity, create an Assign activity. In this Assign activity, map the output variable of the Invoke activity to the result variable that is written back to the client.
13. Compile and deploy the BPEL process on the SOA suite.
14. Use the Oracle BPEL Process Manager console to run the BPEL process.

You can use this BPEL process in another BPEL process and submit jobs to the reports server asynchronously from a BPEL process.

Important: For completing steps from 2 to 11 mentioned in Section 7.10.1, see the following documents:

- *Installing and Configuring Oracle SOA Suite and Business Process Management*
- *Developing SOA Applications with Oracle SOA Suite*

7.10.2 Submitting Jobs to the Reports Server from a BPEL Process Asynchronously

To submit jobs to the Oracle Reports server from a BPEL process asynchronously, do the following:

1. Create a new asynchronous BPEL process project and use the `RWWebservice.xsd` file to define the input and output parameters of the BPEL process.
2. Create a Partner link that refers to the WSDL of the newly created BPEL process:
   - When prompted by Oracle JDeveloper, click **`Yes`** to create Partner link types.
   - Select Partner Role.
3. Create a scope in your process.
4. Within the scope, create an Invoke activity and use it to invoke the runJob operation on the intermediate process Partner link.
5. Create input variables automatically. The input parameter provides input to the runJob operation of the intermediate BPEL process.
6. Create a Receive activity to receive the callback from the intermediate process.
7. Within the scope, before the Invoke activity, create an Assign activity:
   - In this Assign activity, map the user input parameters to the parameters that the intermediate BPEL process requires.
   - Set Param0 = reports job command (string).
   - Set Param1 = `true` (boolean, specifying that the job submission must be synchronous).
8. Within the scope, after the Invoke activity, create a Receive activity.
9. Create input variables automatically. The input parameter receives a response from the runJobresponse operation of the intermediate BPEL process.

10. Within the scope, after the Invoke activity, create an Assign activity. In this Assign activity, map the input variable of the Receive activity to the result variable that is written back to the client.
11. Compile and deploy the BPEL process on the SOA suite.
12. Use the Oracle BPEL Process Manager console to run the BPEL process.

7.11 Optimizing the Deployment of Reports

Before you deploy a report on a machine that is either slow or is running on a load, you may want to configure the following:

- The default timeout period is `150`. This period is calculated from: `ping timeout`, `ping interval`, and `number of retries`. The default values for these are:

```
ping timeout = 30 seconds
ping interval = 20 seconds
number of retries = 3
```

> **Note:** The `number of retries` is applicable only when OPMN successfully connects to Standalone Reports Server and receives regular ONS notifications from the process.

Based on these values, there will be three ping attempts with a timeout of 30 seconds each at 20 second intervals. The first ping is done after the specified `ping interval`. Thus, from the time the Standalone Reports Server is started by OPMN, approximately 150 (20 + 3*30 + 2*20) seconds will elapse before the process is considered unresponsive and restarted. However, if after OPMN connects to Standalone Reports Server but server is too slow in sending regular ONS notifications, then the 30 second timeout applies.

You can configure the ping timeout by adding a ping entry with sufficient timeout configured to the machine's load in following element in `opmn.xml`:

```
<ias-component id="<reports_server_name>">
...
...
<restart timeout="720" retry="2" />
...
<ping timeout="110" interval="30" />
...
```

- User can use these parameters to run reports server process control

`$DOMAIN_HOME/system_components/ReportsServerComponent/<reports_server_name>/data/nodemanager/startup.properties`

RestartMax - The number of times Node Manager can attempt to restart the server.

RestartDelaySeconds - The number of seconds Node Manager should wait before attempting to restart the server.

RestartInterval - The amount of time Node Manager will spend attempting to restart a failed server. Within this period of time Node Manager will attempt to restart the failed server up to the number defined by RestartMax. By default, Node Manager will attempt to restart a server indefinitely until the FAILED_NOT_RESTARTABLE state is reached.

$DOMAIN_HOME/system_components/ReportsServerComponent/<reports_server_name>/data/nodemanager/reports_process.xml

stop timeout - timeout after which reports server will be forcefully shutdown

- Callback timeout (Reports Server-side): Callback timeout is the measure that Reports Server uses to determine the time that it must wait for a response from the engine before timing out. You can specify this value in the rwserver.conf file. This time out period is in milliseconds.

  For example:

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="1"
  maxEngine="1" minEngine="0" engLife="50" maxIdle="30"
  callbackTimeOut="90000">
```

> **Note:** Increase the callbackTimeOut value when the machine is very slow.

7.12 Sample system-jazn-data.xml File

The system-jazn-data.xml is an XML file which is configured by the user to use as an ID store and/or policy store. The file is located in $DOMAIN_HOME/config/fmwconfig.

Sample system-jazn-data.xml file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<jazn-data>
  <jazn-realm>
    <realm>
      <name>jazn.com</name>
        <users>
          <user>
            <name>weblogic</name>
            <guid>23AAB190021911DDBF86C74F01C202FB</guid>
            <credentials>PN0Qr+/dpDRV+jSWP378EdjxWDS0PuAs=</credentials>
          </user>
          </users>
      </realm>
    </jazn-realm>
    <policy-store>
      <applications>
        <application>
          <name>reports</name>
            <app-roles>
              <app-role>
                <name>rw_administrator</name>
                <display-name>Reports Administrator</display-name>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
                <members>
                  <member>

<class>oracle.security.jps.internal.core.principals.JpsXmlUserImpl</class>
                    <name>weblogic</name>
                  </member>
                  <member>
                    <class> weblogic.security.principal.WLSUserImpl </class>
                    <name>weblogic</name>
                  </member>
```

```
                </members>
              </app-role>
              <app-role>
                <name>rw_operator</name>
                <display-name>Reports Operator</display-name>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
              </app-role>
              <app-role>
                <name>rw_monitor</name>
                <display-name>Reports Monitor</display-name>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
              </app-role>
            </app-roles>
            <jazn-policy>
              <grant>
              <grantee>
                <principals>
                  <principal>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
                    <name>rw_administrator</name>
                  </principal>
                </principals>
              </grantee>
              <permissions>
                <permission>
                  <class>oracle.reports.server.ReportsPermission</class>
                  <name>report=* server=* destype=* desformat=*
allowcustomargs=true</name>
                  <actions>*</actions>
                </permission>
                <permission>
                  <class>oracle.reports.server.WebCommandPermission</class>
                  <name>webcommands=* server=*</name>
                  <actions>execute</actions>
                </permission>
              </permissions>
            </grant>
            <grant>
              <grantee>
                <principals>
                  <principal>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
                    <name>RW_BASIC_USER</name>
                  </principal>
                </principals>
              </grantee>
              <permissions>
                <permission>
                  <class>oracle.reports.server.ReportsPermission</class>
                  <name>report=test.rdf server=* destype=* desformat=*
allowcustomargs=true</name>
                  <actions>*</actions>
                </permission>
                <permission>
                  <class>oracle.reports.server.WebCommandPermission</class>
                  <name>webcommands=showmyjobs,getjobid,showjobid server=*</name>
```

```
                <actions>execute</actions>
              </permission>
            </permissions>
          </grant>
        </jazn-policy>
      </application>
</applications>
</policy-store>
```

7.13 Configuring Reports Managed Server

By default, the Oracle Reports application is deployed on a Managed Server.

You can modify the start-up properties of a Reports Managed Server through the `setStartupEnv.sh.sh` file or the WebLogic Server Administration Console. However, some properties like `-Xmx and -Xms`, which are already defined in the `setStartupEnv.sh` (Unix) `setStartupEnv.bat(windows)` file would take precedence and the changes made through the WebLogic Server Administration Console do not take effect. Also, the start-up properties of WLS_REPORTS that are modified using the WebLogic Server Administration Console do not take effect when WLS_REPORTS is started using `startManagedServer.sh`. Hence it is recommended that you use `setStartupEnv.sh` (for UNIX) or `setStartupEnv.bat` (for Windows) to modify the start-up properties for Managed Servers.

You can modify the system or runtime properties for a Managed Server using the `setStartupEnv.sh` (for UNIX) or `setStartupEnv.bat` (for Windows) which is located in `$DOMAIN_HOME/bin`.

You can modify the runtime properties using the `SERVER_MEM_ARGS_64` variable inside the `[ if ["${STARTUP_GROUP}" = "REPORTS-APP-SERVERS"]` block.

You must add any new environment variable for `WLS_REPORTS` Managed Server inside the `[ if ["${STARTUP_GROUP}" = "REPORTS-APP-SERVERS"]` block in the `SetDomainEnv.sh` script.

As the default setting is done for all components that use `setDomainEnv.sh`, the `-Xmx and -Xms` settings are added multiple times to the command line of a Managed Server. Hence you must add the `-Xmx and -Xms` settings at the end of `EXTRA_JAVA_PROPERTIES` variable in the `[ if["${STARTUP_GROUP}" = "REPORTS-APP-SERVERS"]` block.

For example, consider the following snippet:

```
if [ "${START_GROUP}" = "REPORTS-APP-SERVERS" ] ; then
    EXTRA_JAVA_PROPERTIES="-Xms256m Xmx512m ... -Doracle.home=/fmwhome/as1 ...
${EXTRA_JAVA_PROPERTIES}"
```

If you want to add the `-Djobid=random` property and change the `-Xmx and -Xms` settings for WLS_REPORTS to `512m` and `1024m`, you must modify the snippet as follows:

```
if [ "${START_GROUP}" = "REPORTS-APP-SERVERS" ] ; then
    SERVER_MEM_ARGS_64=" -Doracle.home=/fmwhome/as1 ... ${EXTRA_JAVA_PROPERTIES}
-Xms512m -Xmx1024m -Djobid=random"
```

> **Note:** `WLS_REPORTS` is the name of the Managed Server where the Reports application is deployed during installation.

7.14 Enabling HTTPS for Oracle Reports

For enabling HTTPS for Oracle Reports, the **WebLogic Plug-In Enabled** option in the WebLogic Administration Console should be selected for the WLS_REPORTS server by performing the following steps:

1. Log in to the WebLogic Administration Console.
2. If you have not already done so, in the Change Center pane, click **Lock & Edit**.
3. In the Domain Structure pane, expand the **Environment** node, and select **Servers**.

   A list of servers configured in the domain is displayed.
4. Click the **WLS_REPORTS** server.
5. Expand the **Advanced** link near the bottom of the page.
6. Select the **WebLogic Plug-In Enabled** option.
7. Click **Save**.

8

Test to Production

For moving from a Test Environment to Production Environment, follow the steps given in chapter Moving from a Test Environment to a Production Environment available at Oracle *Fusion Middleware Administering Oracle Fusion Middleware* guide.

There are two case:

- Test to Production full
- Test to Production incremental

Table 8–1 Test to Production Use Cases

Use Case ID	Synopsis	Pre-Condition/ Description	Detailed Procedure	Post Condition
Test to Production Full	Administrator creates a complete REPORTS domain in the test environment with the required REPORTS and back-end components. After testing the applications, would like to replicate the entire setup to a production environment	In Test environment, the user has done the following: Installed WLS Installed Forms and Reports Installation Run the WLS config wizard and created reports domain Configured / tuned the reports component instances, wired to different DB etc. Reports does not have a schema but WLS 12c installation requires a RCU schema to be created. Production environment Use T2P scripts to duplicate the bits copyBinary / pasteBinary. Copy domain config using copyConfig / extractMovePlan / pasteConfig. https://docs.oracle.com/middleware/1213/core/ASADM/testprod.htm#ASADM339	See below	The production environment has all the changes in test environment and is working fine
Test to Production Incremental	Administrator has a working production REPORTS setup and would like to test changes in their applications/configuration before rolling them into the production network	Production environment exists Test environment has some patches / config changes that needs to be rolled into the production environment.	See below	The production environment has all the incremental changes and is working fine

For the above cases please find in below tables what is automatically migrated and manual steps needed where required.

Some steps are marked [Optional] and they are applicable only when the relevant functionality is used, for example, Reports plug-ins. In the production environment do the following:

Note that artifacts in domain home are automatically migrated by pasteConfig script. But if user has artifacts outside domain home (for example, used fonts somewhere else in file system) they should be manually migrated

Table 8–2 Test to Production Full

S. No	Test- Production Migration Areas	Automatic/ Manual	Comments/ Steps
1	Reports Component instances created	Automatic	Automatically migrated. Reports Server/Tools/Bridge component instances
2	Reports Configuration	Automatic	Automatically migrated. Reports Server/Tools/Bridge / J2EE App config. See Publishing Reports to Web chapter "**Configuring Oracle Reports Services**"
3	Reports related resources setup	Automatic	Automatically migrated. Fonts / Printer Config /CUPS printer config **Fonts**: Copy any fonts used in test environment from directory specified by environment variable `REPORTS_FONT_DIRECTORY` (default ${DOMAIN_HOME}/reports/fonts) to production environment if it is outside DomainHome.
4	Reports definition files and data tables	Manual	- Copy the reports files (RDF, JSP, REP, XML etc) used in test environment to production environment. - For JSP Web Reports, deploy those JSP reports in production environment reports managed servers in location `${DOMAIN_HOME}/servers/WLS_REPORTS/tmp/_WL_user/reports_<version>/<dirName>/war/` - Migrate Reports specific data tables referred in RDF files to the DB of production environment using DB data migration tools
5	Reports Job related configuration in files system	Automatic	Automatically migrated - Report server Cached Job files, Reports Scheduled job information in files system Note: If absolute path names are used for report names, output names etc while scheduling reports, then path names may not be valid in production environment. In such case user need to reschedule reports in production environment using Reports servlet or Reports command line (rwclient utility).
6	Reports Job related configuration in Database [optional]	Manual	If Job Repository and / or Job Status Repository is configured in Database for Reports server Same schemas need to be created in DB using script ORACLE_HOME/reports/admin/sql/rw_job_repos.sql Migrate data if any in the original DB for schemas `RW_JOBS`, `RW_SERVER_JOB_QUEUE` and `RW_SERVER_QUEUE` to new DB using DB data migration tools.

Table 8–2 (Cont.) Test to Production Full

S. No	Test- Production Migration Areas	Automatic/ Manual	Comments/ Steps
7	Reports OPSS Security related configurations	Automatic	Automatically migrated User and reports server security policy information roles / memberships / policies Credential Store Facility (CSF) entries
8	OID re-association	Manual	If OID is used as for SSOCONN RAD entries they need to be migrated from test OID to production OID using OID data migration tool. Reports need to reassociated to new OID via WLST command
9	DB Proxy users	Manual	Migrate and DB proxy users to production DB using DB cloning tools
10	Reports Plug-in Destinations, Data Sources [optional]	Manual	If Any Reports Plug-in registered, the corresponding jar files needs to be also copied to production environment. The path to these files should also be added in the `REPORTS_CLASSPATH` environment variable in production environment

Note that for configuration change replication in files, it may not be possible to just copy the files from test to production as they have references to OracleHome, DomainHome, Port numbers and other variables. Hence user need to use EM/ EM Mbean browser /JConsole or Manual editing for replication of changes.

- If new reports are created with new fonts in setup, you can follow steps for "Font and printer setup" and incrementally copy new fonts and reports to production environment
- If new CSF keys are created for DB password, you can follow steps for "Reports Security related changes" and incrementally add new CSF keys

Table 8–3 Test to Production Incremental

S. No	Test- Production Migration Areas	Automatic/ Manual	Comments/ Steps
1	Reports Component instances created	Manual	Create Reports Server/Tools/Bridge component instances using WLST commands
2	Reports Configuration	Manual	Merge Reports Server/Tools/Bridge / J2EE App config. See Publishing Reports to Web chapter for a list of config files at **Configuring Oracle Reports Services**
3	Reports related resources setup	Manual	Merge Fonts / Printer Config / CUPS printer config **Fonts**: Copy any fonts used in test environment from directory specified by environment variable REPORTS_FONT_ DIRECTORY (default ${DOMAIN_ HOME}/reports/fonts) to production environment

Table 8–3 (Cont.) Test to Production Incremental

S. No	Test- Production Migration Areas	Automatic/ Manual	Comments/ Steps
4	Reports definition files and data tables	Manual	Copy the reports files (RDF, JSP, REP, XML etc) used in test environment to production environment. For JSP Web Reports, deploy those JSP reports in production environment reports managed servers in location `${DOMAIN_HOME}/servers/WLS_REPORTS/tmp/_WL_user/reports_<version>/<dirName>/war/` Migrate Reports specific data tables referred in RDF files to the DB of production environment using DB data migration tools.
5	Reports Job related configuration in files system	Manual	Copy Report server Cached Job files, Reports Scheduled job information in files system **Note**: If absolute path names are used for report names, output names etc while scheduling reports, then path names may not be valid in production environment. In such case user need to reschedule reports in production environment using Reports servlet or Reports command line (rwclient utility).
6	Reports Job related configuration in Database [optional]	Manual	If Job Repository and/ or Job Status Repository is configured in Database for Reports server Same schemas need to be created in DB using script ORACLE_HOME/reports/admin/sql/rw_job_repos.sql Migrate data if any in the original DB for schemas `RW_JOBS`, `RW_SERVER_JOB_QUEUE` and `RW_SERVER_QUEUE` to new DB using DB data migration tools.
7	Reports OPSS Security related configurations	Manual	Migrate using OPSS command `migrateSecurityStore` User and reports server security policy information roles / memberships / policies Credential Store Facility (CSF) entries
8	OID re-association	Manual	If OID is used as for SSOCONN RAD entries they need to be migrated from test OID to production OID using OID data migration tool. Reports need to reassociated to new OID via WLST command
9	DB Proxy users	Manual	Migrate and DB proxy users to production DB using DB cloning tools

Table 8–3 (Cont.) Test to Production Incremental

S. No	Test- Production Migration Areas	Automatic/ Manual	Comments/ Steps
10	Reports Plug-in Destinations, Data Sources [optional]	Manual	If Any Reports Plug-in registered, the corresponding jar files needs to be also copied to production environment. The path to these files should also be added in the `REPORTS_CLASSPATH` environment variable in production environment

Part III

Managing Runtime Behavior

Part III contains information that will help you to manage the runtime behavior of Oracle Reports:

9

Managing Fonts in Oracle Reports

This chapter provides information about fonts in Oracle Reports:

9.1 Using Fonts

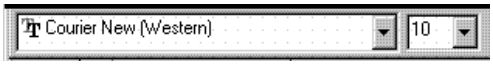
In Oracle Reports, fonts come into play in several areas:

- Fonts in Oracle Reports Builder (at build time)
- Fonts in Report Output (at runtime)
- Fonts in the User Interface

9.1.1 Fonts in Oracle Reports Builder

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

Figure 9–1 Font list in Oracle 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, Oracle 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. Oracle Reports Builder includes only those fonts with an encoding of `iso8859-1`, unless specified differently in the toolkit resource file, `Tk2Motif*fontMapCs`. For more information on `Tk2Motif*fontMapCs`, refer to Section 9.3, "Font Configuration Files".

9.1.2 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 Section 9.3, "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 searched for 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.

With the new font model supported on UNIX in 12*c* Release (12.2.1.3), report output is in most cases the same on UNIX as on Windows, allowing for simplified cross-platform deployment. Oracle Reports reads the font metrics from the appropriate TTF files from the font directory to correctly format the report output. This eliminates the issue of text misalignment due to font metrics mismatches. Fonts for which TTF files are available are found automatically. Note that if a TTF font file is not found, then the font lookup mechanism reverts to the pre-11*g* implementation. With the new font model, there is no change in the builder. The Font Model does a role only during the runtime and not at the design time.

9.1.2.1 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`.
- the TTF font used in the layout objects is used to calculate metrics, which prevents misalignment in the multibyte language report output.

See Also:

- Section 9.3, "Font Configuration Files"
- Section 10.8.1, "ScreenPrinter"
- Section 12.1.1, "Font Lookup"

9.1.3 Fonts in the User Interface

Text in the user interface of Oracle 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.

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 through **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 Oracle 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_LANG is set to UTF8, then you should create an entry like this one:

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

Refer to Section 9.4, "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 Oracle Reports Builder to display undesirable results.

9.2 Adding Fonts

In Oracle Reports, fonts can be added for use:

- At build time (in Oracle Reports Builder)
- At runtime (in the output)
- In the user interface

> **Note:** You must restart the Reports Server after adding fonts to the Reports font directory.

9.2.1 Adding Fonts to Oracle Reports Builder

To build a report in a certain font, the font must be available in Oracle Reports Builder from the font picker when you are designing the report. In order for the fonts to

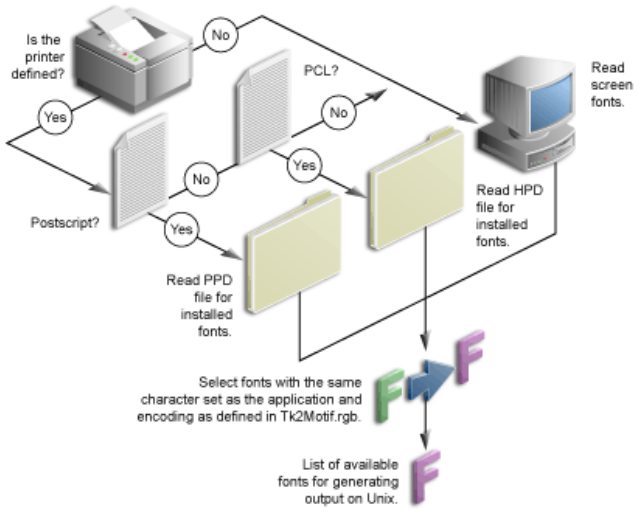
appear in the font picker, the fonts should be added to the system or the display on which Oracle Reports Builder is running. Please review your operating system documentation for adding fonts before attempting this procedure.

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, for example, `t1ascii`.
3. Copy the PFA files to the directory where the fonts have 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, ensure 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 9–2 xlsfonts sample output



5. Start Oracle 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.

9.2.2 Adding 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 Oracle Reports Builder so that you can assign the font to layout objects when designing the report. Refer to Section 9.2.1, "Adding Fonts to Oracle Reports Builder" for further information.

Note: If you use fonts in Oracle Reports Builder that are not available on your runtime platform, you should alias those fonts on the runtime platform. Refer to Section 9.4, "Font Aliasing" for more information.

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

- Adding fonts on UNIX
- Adding fonts on Windows

9.2.2.1 Adding fonts on UNIX

To add TTF fonts:

1. Copy all the TTF and TTC files, which are used in the report, to the REPORTS_FONT_DIRECTORY. The default font directory is `$DOMAIN_HOME/reports/fonts`.
2. Remove any unnecessary aliasing from the uifont.ali file. For example, Arial is aliased to Helvetica, by default. If your report uses the Arial font, you must remove the aliasing from the `uifont.ali` file.

To add PostScript fonts:

1. Copy the AFM file for the new font to *ORACLE_HOME*`/guicommon/tk/admin/AFM`.
2. 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 ...
   ```

   Ensure that the *new_font_name* given in the PPD file is the same as the AFM file, because Oracle Reports searches for this file based on the font name in the PPD file. Also make sure that the AFM file name does not include the `.afm` extension.

   For example, if the AFM file name is `CodedreineunBold`, then the PPD file should contain:

   ```
   *Font CodedreineunBold: Standard "(00.1001)" Standard ROM
   ```

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 Section 9.4, "Font Aliasing".

To add PCL fonts:

In order to use a new font in Oracle Reports, you must have the HPD (printer definition) and TFM files for your printer. The HPD file can be copied from an existing

one. You must 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 must 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`/guicommon/tk/admin/TFM`

3. Modify the alias file, if necessary, as described in Section 9.2.2.1, "Adding fonts on UNIX" for the PostScript printers. The section in which the mapping is done should be `[PCL]`.

9.2.2.2 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 Oracle Reports Builder, the new font will appear in the font picker.

9.3 Font Configuration Files

This section describes all of the files associated with font configuration for Oracle Reports:

- uiprint.txt (UNIX only)
- screenprinter.ppd (UNIX only)
- uifont.ali
- PPD and AFM files (UNIX only)
- HPD and TFM files (UNIX only)
- Tk2Motif.rgb (UNIX only)

> **Note:** In Oracle Reports, configuration of fonts should be done only through Oracle Enterprise Manager. Refer to Chapter 6, "Administering Oracle Reports Services Using Oracle Enterprise Manager", Section 6.4.1, "Configuring Fonts" for information about updating configuration settings through Oracle Enterprise Manager.

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:

- Section 10.3.1, "Editing uiprint.txt File"
- Section 10.4.6, "uiprint.txt"

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. See Section 10.8.1, "ScreenPrinter".

uifont.ali

The `uifont.ali` file is found in the following location on Windows and UNIX:

On Windows: `$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\common`

On UNIX: `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin`

This file contains mapping information for fonts that can be substituted for other fonts at runtime.

Caution: Do not alter the sections as Oracle Reports parses the `uifont.ali` file looking for keywords. The sections can be in any order. Any font configuration should be done only through Oracle Enterprise Manager. Refer to Chapter 6, "Administering Oracle Reports Services Using Oracle Enterprise Manager", Section 6.4.1, "Configuring Fonts"for information about updating configuration settings through Oracle Enterprise Manager.

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

- Font or character set names require double quotation marks around two or more words, or spaces.
- Comments are specified using # in the first column.
- 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 `.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 AFM and PFB files are required in this order).

The general format for aliasing in `uifont.ali` 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, such as `Courier`. The *Style*, *Weight*, *Width*, and *CharSet* may either be a numeric value or a predefined identifier or 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=font_file_name
Face.Size.Style.Weight.Width.CharSet=font_file_name
```

The following is a list of recognized names and their numeric equivalents:

Table 9–1 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 9–2 Weights and Their Numeric Equivalents

Weight Name	Numeric Equivalent
Ultralight	1
Extralight	2
Light	3
Demilight	4
Medium	5
Demibold	6

Table 9–3 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 (Section 9.5.7, "CID Fonts") from the Asian font pack from Adobe. For example, in your Japanese report you have aliased a multibyte `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.

Refer to Section 9.4, "Font Aliasing" for more information.

PPD and AFM files (UNIX only)

PostScript Printer Definition (PPD) files and Adobe Font Metrics (AFM) files are supplied by Adobe and by printer vendors. The PPD files contain information about the printer, and the AFM files contain metrics information of the fonts. 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 Section 9.3.1, "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 that provide Oracle Reports with 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 must do the following:

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 (UNIX only)

PCL (Hewlett-Packard Printer Control Language) uses HPD (Hewlett-Packard Document) and TFM (TEX Font Metrics) 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.

`Tk2Motif*fontMapCs` 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 `NLS_LANG` should be set to `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 (DOMAIN_HOME/config/FRComponent/frcommon/guicommon/tk/admin)` file as comments.

9.3.1 File Searching

The criteria for searching files is dependent upon the type of file and the various environment variables defined.

Table 9–4 File Information

File Name	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 `TK_`*`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 `TK_PPD`, then in `ORACLE_PPD`, and then in `ORACLE_HOME/guicommon/tk/admin/PPD`, and then in `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/PPD`

For example, looking for `uiprint.txt`, Oracle Reports will first look at the environment variable `TK_UNKNOWN`, then look at `ORACLE_UNKNOWN`, and then in the default directory.

9.4 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 Windows to Motif one would use font aliasing to map the Windows Arial to a font available on Motif, such as Helvetica.

The font enhancements in Oracle Reports make font aliasing unnecessary in almost all cases. In prior releases, a report may have been created with fonts that are readily available on Windows, but not on UNIX (for example, Arial font). In such cases, it was necessary to alias the Windows fonts to other fonts with a similar style available on UNIX (for example, Helvetica). Now, with support for TTF and TTC files on UNIX, a font such as Arial is supported on both Windows and UNIX, eliminating the need for aliasing.

This section includes the following topics:

- Specifying Aliasing Information
- Font Aliasing Mechanism
- Font Alias File Sections
- Font Aliasing File Verification

9.4.1 Specifying Aliasing Information

If font aliasing is necessary, use Oracle Enterprise Manager to define the aliasing, instead of directly editing the `uifont.ali` file as in prior releases. For information about using Oracle Enterprise Manager for font configuration, see Section 6.4.1, "Configuring Fonts".

9.4.2 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 Section 9.4.3, "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 must 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 uifont.ali file, Oracle Reports looks for the font following the font lookup procedure, which is described in Section 9.1.2.1, "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.

9.4.3 Font Alias File Sections

The uifont.ali file consists of various sections which contains font mapping instructions for a particular area, as shown in Table 9–5. 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 9–5 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) Used for True Type font subsetting.
`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]`

See Also: "Repairing Fonts Not Appearing Correctly in Web Source View" in Section 9.9, "Troubleshooting Font Issues".

If you want to look at the `uifont.ali` file, it is located in the following directory on Windows and UNIX:

On Windows: `$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\common`

On UNIX: `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin`

The section for font aliasing in the `uifont.ali` file is `[PDF]`, which defines font mappings using the following formats:

- Single byte fonts:

```
[PDF]
"font_name"="font_name"
```

- Multibyte fonts:

```
[PDF]
character_set = "font_name"
```

or

```
"font_name".....character_set="font_name"
```

Here is an example of a font aliasing entry in the `uifont.ali` file:

```
[ PDF ]
```

```
/*Alias TrueType to available Type1 font */
"Kino MT" = UtopiaBold
/*Alias multibyte to available CID font */
.....SJIS = "HeiseiKakuGo-W5-Acro"
```

where:

- `"Kino MT" = UtopiaBold` substitutes every `Kino MT` character found with the `UtopiaBold` equivalent.
- `.....SJIS = "HeiseiKakuGo-W5-Acro"` substitutes every multibyte character set found with the `HeiseiKakuGo-W5-Acro` (CID) equivalent.

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. The `uifont.ali` file is text readable; that is, 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.

> **Note:** Although you can manually edit the uifont.ali file, it is recommended that you use Oracle Enterprise Manager for all font-related configuration tasks.

9.4.4 Font Aliasing File Verification

To verify whether the `uifont.ali` file is correct, you can run the font check utility, which 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:

On Windows:

```
fnchk.exe filename
```

On UNIX:

```
mfontchk filename
```

where *`filename`* is the name of the modified `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 "$DOMAIN_
```

```
HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_
name>/guicommon/tk/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.

9.5 Font Types

This section discusses the fonts and character sets relevant to Oracle Reports:

- Character Sets
- Unicode
- Type1 Fonts
- TrueType Fonts
- TrueType Collection
- Barcode Fonts
- CID Fonts

9.5.1 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 must 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`).

9.5.2 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. For information about using Unicode in your multilingual applications, refer to Section 23.5, "Unicode".

9.5.3 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` and `.pfb`) contain character information, while the metric files (`.afm` (Adobe Font Metric) and `.pfm` (Printer Font Metric)) contain the metric information to form the character. These metrics files are ASCII files with a well-defined easy-to-parse structure.

9.5.4 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.

9.5.5 TrueType Collection

A TrueType Collection (TTC) is an efficient way of sharing common font data, such as character information and glyphs. This data sharing results in an optimized file size as the common glyphs are stored in a single file structure, instead of within each font. The end result is a single file that is a combination of two or more fonts. For example, certain Japanese fonts in a font family may share a common set of kanji characters. They can be included in a TTC file.

For example, the TTC file, `msgothic.ttc`, is a collection file consisting of three fonts. They are MS Gothic, MS PGothic, and MS UI Gothic.

9.5.6 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.

9.5.7 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.

9.6 Verifying Report Output on Different Platforms

Oracle Reports 12*c* Release (12.2.1.3), uses the widely available font formats like TTF and TTC on both Windows and UNIX to generate report output that in most cases looks identical on both platforms, with no configuration necessary.

Oracle Reports reads the font metrics from the appropriate TTF files to correctly format the report output. This eliminates the issue of text misalignment due to font metrics mismatches. Fonts for which TTF files are available are found automatically. Note that if a TTF font file is not found, then the font lookup mechanism reverts to the pre-11*g* implementation.

Run a sample TTC font type report on Windows:

1. Open the TTC font type report in Reports Builder.
2. Select **File > Generate to File > PDF** and save the output.

Run the same TTC font type report on UNIX:

1. First check whether the TTC font used by the report is already available on the UNIX server machine in `$DOMAIN_HOME/reports/fonts` (for example, `MSGOTHIC.TTC`). If it is not available, copy it from the Windows machine (`C:\WINDOWS\Fonts)`.
2. Run the report using the following command line:

   `http://`*`host`*`:`*`port`*`/report=`*`report_name`*`.rdf&destype=cache&desformat=pdf&userid=`*`user`*`/`*`password`*`@`*`db`*

Compare the output on UNIX with that on Windows to confirm that they are almost identical.

> **Tip:** On UNIX, set the DPI as in Windows. For example, if you change the UNIX DPI from 96 to 600, which is the Windows DPI value, the PDF and RTF outputs on UNIX will be identical to that of Windows.

However, the HTML ouput file is large because the DPI value of the screen printer is changed. This problem occurs only if a valid printer is not set and the screen resolution (screenprinter.ppd) is used for both PDF and HTML drivers. For example, if the valid/dummy printer is set to TK_PRINTER with the DPI resolution 600, the same value as that of Windows, the HTML and PDF outputs on UNIX will be identical to that of Windows.

Example

Here is an example of how to produce report HTML output that looks the same on Windows and UNIX if the DPI of Windows is 600:

1. If you do not have a valid printer, add the following entry in the uiprint.txt file:

   Font:ASCII: DPI changed PPD file:default.ppd:

2. Change the Default Resolution to 600 in the datap462.ppd file as follows:

   *DefaultResolution: 600dpi

> **Note:** Do not change the resolution of screenprinter.ppd.

3. If you have a valid printer, configure the printer in the uiprint.txt file and change the DefaultResolution to 600.
4. Run the report after placing the fonts in the font folder.

9.7 Running a Unicode Report using TTF/TTC Fonts

Oracle Reports 12*c* Release (12.2.1.3), uses the new font mechanism to run a Unicode report on UNIX using the TTF / TTC fonts, generating report output that in most cases looks identical on both platforms.

Run a sample Unicode report on Windows:

1. Navigate to the EM MBean browser Weblogic Domain > System MBean Browser

   Navigate to reports tools mbean for uifont.ali
   `oracle.frcommon.config:type=uifont.ali,name=uifont-<componentName>`

   - Add the following entry under the [PDF:Subsetting] section:

     ```
     "Arial Unicode MS"="ARIALUNI.TTF"
     ```

   - Ensure there is no aliasing defined for Arial in the Global section. If there is, comment it out:

     ```
     #Arial=Helvetica
     ```

2. Add the font `ARIALUNI.TTF` to `REPORTS_PATH`.
3. Set `NLS_LANG` in the Windows registry to `AMERICAN_AMERICA.UTF8`.
4. Open the sample report in Reports Builder.
5. Select **File > Generate to File > PDF**.
6. Save the report output.

Run the same Unicode report on UNIX:

1. Check whether the Unicode font used by the report is already available on the UNIX server machine in `$DOMAIN_HOME/reports/fonts` and in the `REPORTS_PATH` (for example, `ARIALUNI.TTF`). If it is not available, copy it from the Windows machine to `$DOMAIN_HOME/reports/fonts`.
2. Set `NLS_LANG` in `reports.sh` to `AMERICAN_AMERICA.UTF8`.
3. Run the report using the following command line:

   ```
   http://host:port/report=report_
   name.rdf&destype=cache&desformat=pdf&userid=user/password@db
   ```

   For example:

   ```
   http://host:port/report=Unicode.rdf&destype=cache&desformat=pdf&P_LANG_
   ID=JA&userid=oe/oe@mydb
   ```

Compare the output on UNIX with that on Windows to confirm that they are identical.

9.8 Diagnosing Font Issues

To diagnose font usage and issues, you can view the log files in either of the following ways:

- Using the Command Line

- Using Oracle Enterprise Manager

9.8.1 Using the Command Line

To view the log files using the command line:

- Open the diagnostics log file:

  ```
  $DOMAIN_HOME/servers/<reports_server_name>/logs/rwEng-id_diagnostics.log
  ```

- Notice that the font used for each object in the report is logged without any ambiguity. If a TTF font file was not found, it is logged and noted that the pre-11*g* mechanism was used for finding the appropriate font, which is also logged.

9.8.2 Using Oracle Enterprise Manager

Refer to Section 6.6.1, "Viewing and Searching Log Files" in Chapter 6, "Administering Oracle Reports Services Using Oracle Enterprise Manager"

9.9 Troubleshooting Font Issues

To help resolve font issues that may occur in your applications, this section provides the following troubleshooting information:

- Checking Whether the Desired Font Is Used in a PostScript File
- Creating Output
- Reading the Output File
- Verifying the Output File
- Correcting Printed Font
- Checking Environment Variables
- Repairing Fonts Not Appearing Correctly in Web Source View
- Understanding Limitations
- Resolving Common Problems

Checking 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 information 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 9–6 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 9–7 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.

Reading 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 preceding example, 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....
```

Correcting Printed Font

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, Oracle Reports did not generate the code to the output file. Check for the HPD and TFM files.

Checking 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.

Note: Set the environment variable `DEBUG_SLFIND` to any file name and run the report. The debug information is written in that particular file.

Usage: `# setenv DEBUG_SLFIND mydebug.txt`

See Appendix B, "Environment Variables".

Repairing Fonts Not Appearing Correctly in Web Source View

Text in the user interface of Oracle 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 Oracle 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 My Oracle Support (http://support.oracle.com), patch ID 2766564.

Understanding Limitations

On Windows:

- For Unicode, Windows provides True Type Big Fonts. These fonts contain the characters necessary to display or print text from more than one language. For example, if you try to type, display, or print Western European, Central European, and Arabic text on a field and see unexpected characters, then you are probably not using a Big Font. Big Fonts for single-byte languages provided by Microsoft Windows are Arial, Courier New, and Times New Roman. See http://www.microsoft.com/typography/fonts/default.aspx.
- Wingdings fonts may not appear when `NLS_LANG` is set to `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. Oracle 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.

Resolving 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 setting the Horizontal Elasticity property 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 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. Close Oracle Reports Builder and 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 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 Oracle 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 Oracle 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 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 UNIX 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 UNIX. 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 ensure that `uifont.ali` is in the correct directory.

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 must 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 ensure 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 -`.

10

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
- Enhanced Printing on Linux Using CUPS
- Globalization Support
- Debugging Options
- Removing DISPLAY and Printer Dependencies on UNIX
- Frequently Asked Questions

10.1 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

10.1.1 General Printing Mechanism

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; that is, you must install a specific printer driver for a printer. These printer drivers know how to communicate to the printer and provide services to

applications that send output to the printer. Applications can access the printer properties, change their properties, and perform printing through these standard APIs.

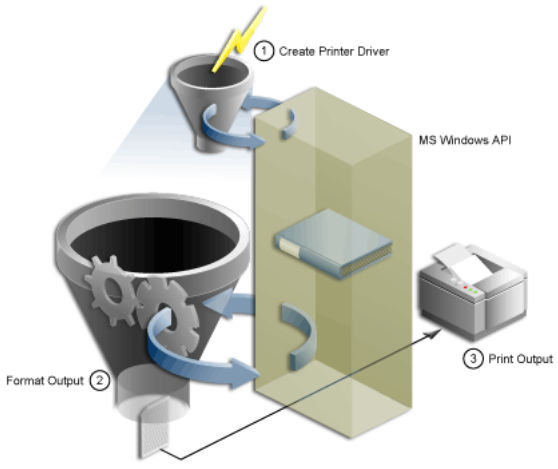
Motif and character-based UNIX operating systems do not have their own standard interface to printers like Microsoft Windows. Individual applications are responsible for sending their output in a streamed file to the printer and adhering to the specifications of the printer. On UNIX platforms, Oracle Reports output must be formatted properly (for example, 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.

10.1.2 Oracle Reports Printing Mechanism on UNIX and Windows

Figure 10–1 and Figure 10–2 depict the differences between Oracle Reports printing on UNIX and on Microsoft Windows.

Figure 10–1 Oracle Reports printing on UNIX

Figure 10–2 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 (for example, 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 through the printing executable (for example, `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 must 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 (for example, 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.

10.1.3 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 these versions.

10.2 Setting Up a Printer on UNIX

This section describes:

- Installing a Printer on UNIX
- Verifying the Printer Setup for Oracle Reports

10.2.1 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.

```
#!/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"
```

10.2.2 Verifying the Printer Setup for Oracle Reports

To verify that your printer queue installed correctly:

1. Ensure that the PPD or HPD file used with the installed printer queue is available in the following location:

   ```
   $DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_
   name>/guicommon/tk/admin/PPD
   $DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_
   name>/guicommon/tk/admin/HPD
   ```

2. Ensure that the font metrics, AFM or TFM files, installed on the printer are available in the following location:

   ORACLE_HOME`/guicommon/tk/admin/AFM`
   ORACLE_HOME`/guicommon/tk/admin/TFM`

10.3 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

10.3.1 Editing uiprint.txt File

As discussed in Section 10.1, "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:

```
$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_
name>/guicommon/tk/admin/uiprint.txt
```

`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 10–1 describes each element of the `uiprint.txt` entry.

Table 10–1 `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 10–1 (Cont.) `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: ■ 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 the following locations serially: ■ `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/PPD` ■ `ORACLE_HOME/guicommon/tk/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*`/guicommon/tk/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: Section 10.3.2, "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 must 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:
```

10.3.2 Environment Variables

This section lists the environment variables related to printing:

See Also: Appendix B, "Environment Variables" for more information on the environment variables that can be set in Oracle Reports.

- TK_PRINTER / PRINTER
- TK_PRINTER
- TK_PRINT_STATUS
- REPORTS_NO_DUMMY_PRINTER
- TK_HPD and ORACLE_HPD
- TK_PPD and ORACLE_PPD
- TK_TFM and ORACLE_TFM
- TK_AFM and ORACLE_AFM

10.3.3 Print Property Dialog Boxes

On UNIX, Oracle Reports Builder provides several dialog boxes for printer-related operations.

10.3.3.1 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, (for example, scale, rotation, width, and height) for formatting a page on a printer.

10.3.3.2 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.

10.4 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

10.4.1 Overview of Files

Table 10–2 lists files used by Oracle Reports for printing on UNIX.

Table 10–2 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

10.4.2 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 (for example, 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 10–3 shows some examples of PPD files that are shipped with Oracle Reports:

Table 10–3 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

Table 10–3 (Cont.) Common PPD Files Shipped with Oracle Reports

PPD File Name	Corresponding Printer
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.

10.4.2.1 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.

10.4.3 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.

10.4.4 Font Metrics Files

Oracle Reports supports two kinds of font metrics files:

- AFM files
- TFM files

10.4.4.1 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 Section 10.2.2, "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.

10.4.4.2 TFM files

PCL 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, after 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 Section 10.2.2, "Verifying the Printer Setup for Oracle Reports".

10.4.5 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:

On Windows: `$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\common`

On UNIX: `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/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; that is, printer or system.

> **See Also:** Chapter 9, "Managing Fonts in Oracle Reports" for more font-related information.

10.4.6 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:** Section 10.3.1, "Editing uiprint.txt File" for more information about `uiprint.txt`.

10.4.7 Editing the Printer-Related Files

This section describes how to edit the various print-related files:

- Editing PPD files
- Editing HPD files for PCL printing

10.4.7.1 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 want to change:

- Changing the default paper size
- Changing the printer margin settings
- Adding a new font entry to PPD files
- Overriding the printer tray setting

10.4.7.1.1 Changing the default paper size Suppose that you need the page size to be A4 for some of your reports. On UNIX 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
```

10.4.7.1.2 Changing the printer margin settings To change the margins, you must 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 10–4 `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
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.

10.4.7.1.3 Adding a new font entry to PPD files On PostScript printers, Oracle Reports only enables you to use fonts known to be available on the printer. Since printers are rarely available for personal requests on multiprocess 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 want 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
```

10.4.7.1.4 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.

10.4.7.2 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 want to change:

- Changing the paper size
- Adding a new font entry

10.4.7.2.1 Changing the paper size For example, to change the papersize to A4, add the following to the HPD file used:

```
<defaultpaper=A4>
```

10.4.7.2.2 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.

10.5 Enhanced Printing on Linux Using CUPS

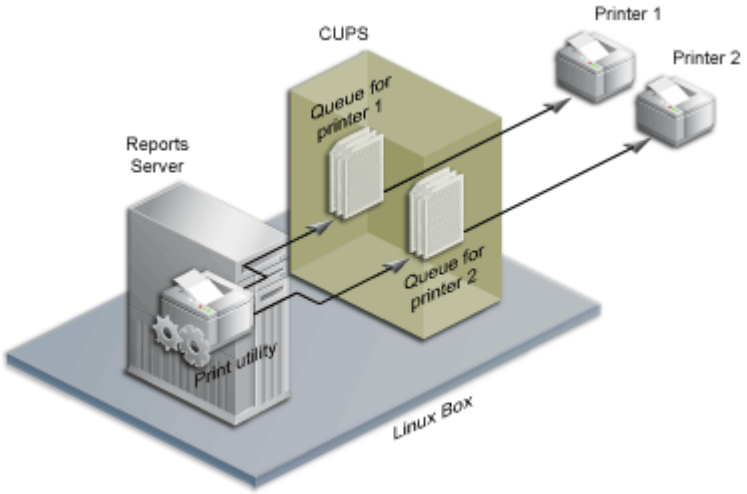
Common UNIX Printing System (CUPS) is the default printing system on most Linux distributions. This section describes how to set up CUPS for printing from Reports Server.

CUPS printing is disabled by default. To enable CUPS printing, set the environment variable `REPORTS_CUPS_PRINTING` to `YES`. See Appendix B.1.40, "REPORTS_CUPS_PRINTING".

The primary advantage of CUPS is that it is a standard and modularized printing system that can process numerous data formats on the print server and also supports Internet Printing Protocol (IPP). With this feature, it is possible to directly print PDF files from Reports Server and it also simplifies network printing.

Figure 10–3 shows how Oracle Reports interacts with CUPS.

Figure 10–3 Interaction Between Oracle Reports and CUPS Server



With this configuration, printers must be configured on all the CUPS running on all the machines where Reports Server is running.

Several text files are used to configure CUPS. For more configuration-related information, see the *CUPS Software Administrators Manual* at `http://www.cups.org/doc-1.1/sam.html#CONTENTS`.

By default, CUPS does not allow access from other network machines. To configure CUPS to allow access from remote machines, perform the following steps:

1. Enter the following command to open a CUPS configuration file:

   ```
   open /etc/cups/cupsd.conf
   ```

2. Add a `Listen` instruction, as follows:

   a. Scroll to the bottom of the configuration file where the other `Listen` instructions are declared.

   b. Copy `Listen 127.0.0.1:631` and paste it above or below the original.

   c. Replace `127.0.0.1` with the Linux server's IP address.

3. Configure each printer, as follows:

   a. In the configuration file, locate:

```
<Location /printers/your_printer_queue>
```

b. Comment the instruction `Deny From All`.

c. Change `Allow from 127.0.0.1` to `Allow from All`.

d. Repeat for all printer or fax queues that you wan to make accessible.

4. Save the configuration file and restart CUPS.

   - To stop CUPS, enter the following command:

     ```
     /etc/rc.d/init.d/cups stop
     ```

   - To start CUPS, enter the following command:

     ```
     /etc/rc.d/init.d/cups start
     ```

The Red Hat Advanced Server provides a configuration wizard to help you set up your printers. To use the configuration wizard for this task, perform the following steps:

1. While logged on as the root user, open a command prompt and enter:

   ```
   redhat-config-printer
   ```

   to display the Printer configuration window.

2. Select the **New** tab to launch the Add a new print queue wizard.
3. Follow the wizard prompts to:
   - Enter a queue name.
   - Select the queue type.
   - Select the printer model.
4. Review your selections and click **Apply** to create your new print queue.
5. Test your printer on CUPS, as follows:
   - Launch a browser on RedHat with the URL `http://localhost:631`.
   - Select the **Printers** tab. The printer you just created will be listed.
   - Click **Print Test Page** to test your printer setup. If the test page does not print, repeat the configuration steps. Ensure that your printer type and model selections are correct.

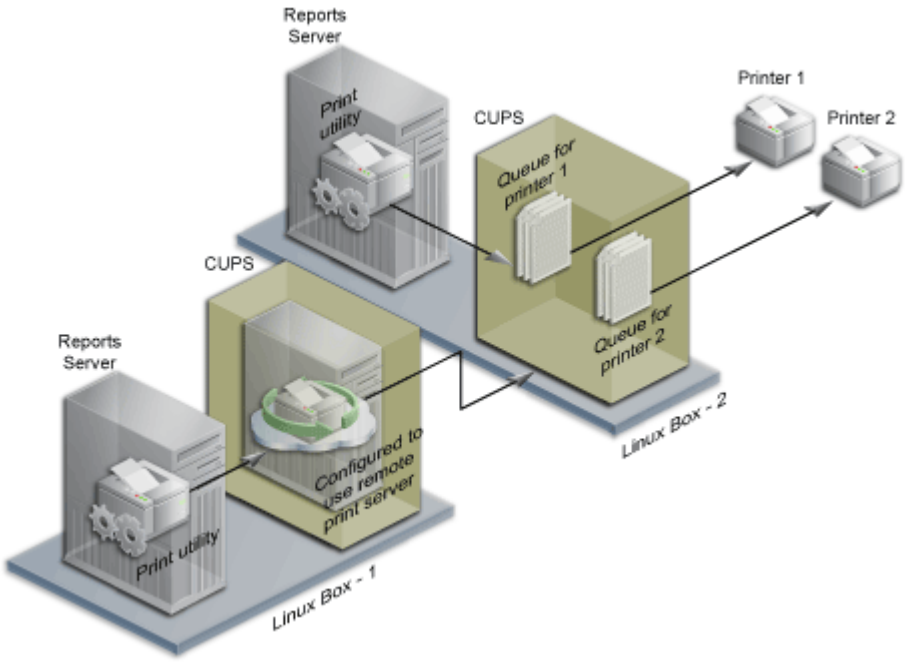
Note: `lpadmin` can also be used to add a printer from the command line. Refer to the Linux man page for this command for more details.

10.5.1 Setting Up a Single Server for Printing

The default CUPS configuration is to use `localhost` as the print server. To make the CUPS use a remote server, you must change the server name in the `/etc/cups/client.conf` file.

Figure 10–4 illustrates adding printers to a single CUPS server, and configuring all other machines running Reports Server to route their print requests to the remote CUPS server.

Figure 10–4 Routing Print Requests to a Single Remote CUPS Server



For information about the various other printing models, refer to the *CUPS Software Administrators Manual* at `http://www.cups.org/doc-1.1/sam.html#CONTENTS`

Note: When Oracle Reports adds a printer to the CUPS server, it assigns a printer name, which is the name that must be used when referring that printer. Internally, the name is translated to the proper call depending on the device URI used in the configuration.

10.6 Globalization Support

This section explains multibyte character set printing support in Oracle Reports. It also explains the IX and PASTA utilities, which are supported only for Oracle Reports when installed and used in conjunction with Oracle Applications.

- Multibyte Character Set Printing
- Overview of IX and PASTA

10.6.1 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 (`desformat=pdf`), 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 9, "Managing Fonts in Oracle Reports" and Chapter 11, "Using PDF 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.

10.6.2 Overview of IX and PASTA

When installed and used with Oracle Applications, Oracle Reports includes utilities for font embedding in PostScript output.

For character-mode reports, the utility is called PASTA. For bit-mapped reports, the utility IX enables 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 through 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 UNIX 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. You can also get this information at My Oracle Support (`http://support.oracle.com`), patch ID 189708.1 and patch ID 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.

10.7 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

10.7.1 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 TK_PPD and TK_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 TK_PPD(8).
slsfindfile(): environment variable not set
slsfindfile(): checking environment variable ORACLE_PPD(10).
slsfindfile(): environment variable not set
slfpath(): looking up path
/oraclehome/guicommon/tk/admin/PPD/
slfexist(): testing /oraclehome/guicommon/tk/admin/PPD
slfexist(): testing /oraclehome/guicommon/tk/admin/PPD/default.ppd
slsfindfile():returned
/oraclehome/guicommon/tk/admin/PPD/default.ppd
slfindfile(): type = 39 (AFM)slfindfile(): name = Courier-Bold
slsfindfile(): checking environment variable TK_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 /oraclehome (len=18)
slfpath(): looking up path/oraclehome/guicommon/tk/admin/AFM/
slfexist(): testing /oraclehome/guicommon/tk/admin/AFM
slfexist(): testing /oraclehome/guicommon/tk/admin/AFM/Courier-Bold
slsfindfile():returned /oraclehome/guicommon/tk/admin/AFM/Courier-Bold
slfindfile(): name = uiprint.txt
slsfindfile(): checking ORACLE_HOME environment variable.
slfpath(): looking up path/oraclehome/guicommon/tk/admin/
slfexist(): testing /oraclehome/guicommon/tk/admin
slfexist(): testing /oraclehome/guicommon/tk/admin/uiprint.txt
slsfindfile(): returned /oraclehome/guicommon/tk/admin/uiprint.txt
```

10.7.2 TK_DEBUG_POSTSCRIPT

This variable effects the PostScript output generated by Oracle Reports. Table 10–5 shows the settings for this variable.

Table 10–5 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 need not provide this output to Oracle Support for investigation.

10.8 Removing DISPLAY and Printer Dependencies on UNIX

Prior to Oracle Reports 10*g* Release 1 (9.0.4) on UNIX, you had to set the `DISPLAY` environment variable in order for Reports Server to use the windowing system display surface for creating images and getting pixel resolution. This dependency was removed with Oracle Reports 10*g*.

Additionally, earlier releases required a valid printer on UNIX for fonts. When no valid printer was available, Oracle Reports Services used the screen fonts, which again required setting the `DISPLAY` environment variable. Now, Oracle 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, Oracle Reports Services no longer requires a printer on UNIX.

By default, the environment variable `REPORTS_DEFAULT_DISPLAY` is set to `YES`, which specifies that Oracle 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 dependency on the `DISPLAY` environment variable as in releases prior to 10*g* Release 1 (9.0.4), you can set `REPORTS_DEFAULT_DISPLAY=NO`.

See Also: Section B.1.43, "REPORTS_DEFAULT_DISPLAY"

10.8.1 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 : $DOMAIN_
HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_
name>/guicommon/tk/admin
screenprinter.ppd : $DOMAIN_
HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_
name>/guicommon/tk/admin/PPD
```

Note: Configuration of `screenprinter.ppd` should be done only through Oracle Enterprise Manager. Refer to Chapter 6, "Administering Oracle Reports Services Using Oracle Enterprise Manager", Section 6.4.1, "Configuring Fonts"for information about updating configuration settings through Oracle Enterprise Manager.

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 metrics 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 multibyte 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:

- Chapter 9, "Managing Fonts in Oracle Reports"

10.8.2 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 10–6 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_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 enables you to import images of these same formats into your report.

Usage Notes

- UNIX only: Enabling advanced imaging means that you can no longer use the old Computer Graphics Metafile (CGM) and Oracle Graphics Data (OGD) formats in HTML or HTMLCSS output. If you require these formats for input sources, you should set `REPORTS_DEFAULT_DISPLAY=NO`. This limitation does not apply on the Windows platform.
- Running a report with JPEG images (`REPORTS_OUTPUTIMAGEFORMAT=JPEG`) to RTF output causes an increase in the RTF file size that is not directly proportionate to the image size. This occurs because the binary image stream is first converted to HEX characters and then written to RTF. This conversion increases the file size. This is consistent with the RTF specification and is expected behavior. However, an RTF file with JPEG images is of a smaller size when compared to an RTF file with BMP images.

See Also:

- Section A.7.3, "OUTPUTIMAGEFORMAT"
- Section B.1.43, "REPORTS_DEFAULT_DISPLAY"
- Section B.1.60, "REPORTS_OUTPUTIMAGEFORMAT"

10.9 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

> **Note:** Reports Server uses `rwlpr` for submitting a print job. For `rwlpr` logging for Windows, when you enable tracing for Reports Server using either `traceModule=all` or `traceModule=server`, a printing diagnostic log (*`server_name`*`-rwlpr-`*`jobid`*`.log`) is created in the log directory (`$DOMAIN_HOME/servers/<reports_server_name>/logs/`) for `destype=printer`. This log file will contain information regarding the messages that can be used to diagnose any printing issues, such as spooler problem.

10.9.1 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.

Action:
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:

  `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/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 Appendix B, "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.

Action:

The DESFORMAT parameter specifies which output format is needed. Valid formats are:

- For bitmapped reports, any of the output formats supported by Oracle Reports (PostScript, PCL, PDF, HTML, XML, HTMLCSS, ENHANCEDSPREADSHEET, SPREADSHEET) 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, ensure that you change the MODE parameter to Character and use any valid .PRT file.

Table 10–7 maps the command line options (DESTYPE, DESNAME, and DESFORMAT) to the printer by what you are trying to achieve.

Table 10–7 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.

Action:

1. Verify that there is a valid entry in uiprint.txt.

2. If you have multiple printer queue entries in `uiprint.txt` and you want 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 on printer-related environment variables, refer to Appendix B, "Environment Variables".

If there is no printer available for your system, refer to Section 10.3, "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.

Action:

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_INSTANCE/config/reports/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
```

Printing on Solaris 2.9

If you print a report using the DESTYPE=PRINTER and the DESNAME=*printer_name* command line options on Solaris 2.9, you will encounter the following errors:

```
REP-0069: Internal error
REP-57054: Inprocess job terminated with error
REP-50157: Error while sending file to printer 2op837a.Exit with error code 1
```

To resolve this issue, you must do the following:

> **Note:** Create a backup of the rwlpr.sh file before proceeding. On Solaris, rwlpr.sh is the printing script file located in the $ORACLE_INSTANCE/config/reports/bin directory. This script file supports lp and lpr commands by default.

1. Navigate to the following line at the end of the file:

   ```
   #either LPR or LP Command was found
   ```

2. Add an OR operator to the existing if...else condition.

   ```
   if [ `uname -r` = "5.8" ] || [ `uname -r` = "5.9" ] # If Solaris Release 5.8 /
   5.9
      ...
      else
      # parse and Fix the command Line as Required by lp
      ...
        if [ `uname -r` = "5.8" ] || [ `uname -r` = "5.9" ]# If Solaris Release
        5.8/ 5.9
   ```

3. The if...else if condition checks for the Solaris Release version. Based on the version number, it strips the quotes from the printer name and passes it to the print command.

10.9.2 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.

10.9.3 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 Appendix B, "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 procedure. 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 enables you to easily print pages of the same report on different sheets of paper.

Note: For a description of the SRW built-in package, including the SRW.SET_PRINTER_TRAY built-in procedure, see the *Oracle Reports online Help*.

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 must 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?

10.9.4 Font-Related Printing Issues

See Also:

- Chapter 9, "Managing Fonts in Oracle Reports" for more font-related information.
- Chapter 12, "Font Model and Cross-Platform Deployment" for resolving cross-platform font 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.

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?

Prior to Oracle Reports 10*g* on UNIX, you had to set the `DISPLAY` environment variable in order for Reports Server to use the windowing system display surface for creating images and getting pixel resolution. This dependency is removed with Oracle Reports 10*g*.

Additionally, earlier releases required a valid printer on UNIX for fonts. When no valid printer was available, Oracle Reports Services used the screen fonts, which again required setting the `DISPLAY` environment variable. Now, Oracle 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, Oracle Reports Services no longer requires a printer on UNIX.

See Also: Section 10.8, "Removing DISPLAY and Printer Dependencies on UNIX".

10.9.5 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 Oracle 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 Oracle Reports Builder.
  2. Go to the Paper Layout view.
  3. Click 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 Oracle 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 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 9, "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.

11

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:

11.1 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:

11.1.1 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.

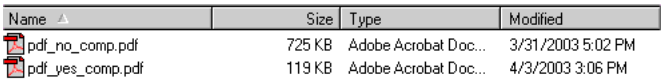
Compressed files are about one fifth the size of the original file. Testing has shown that the best case compression ratio of one-eighth to the worst case compression ratio of one-half was achieved based on the contents in the original file.

11.1.1.1 Setup

By default, PDF output generated by Oracle Reports is compressed. To specify the level of compression, use `PDFCOMP` on the command line. See Section A.7.24, "PDFCOMP".

Although compressed files download quickly, the time taken to generate a compressed file is much more when compared to a non-compressed file.

Figure 11–1 Compressed Output Versus 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.

11.1.2 Font-Related Features

This section outlines the PDF font-related features supported by Oracle Reports:

- Font Aliasing
- Font Subsetting
- Font Embedding
- Font Feature Summary

11.1.2.1 Font Aliasing

Font aliasing enables you to substitute one font for another; that is, 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. For font aliasing considerations when designing multilingual applications, see Section 23.2.1.3.2, "Font Aliasing Considerations".

The font enhancements in Oracle Reports make font aliasing unnecessary in almost all cases. In prior releases, a report may have been created with fonts that are readily available on Windows, but not on UNIX (for example, Arial font). In such cases, it was necessary to alias the Windows fonts to other fonts with a similar style available on UNIX (for example, Helvetica). Now, with support for TTF and TTC files on UNIX, a font such as Arial is supported on both Windows and UNIX, eliminating the need for aliasing.

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 listed in Table 11–1, which are available as a free download from Adobe.
- Type1 PostScript fonts.
- TrueType fonts.

Table 11–1 outlines the mapping between Oracle `NLS_CHARACTERSET`, CMap name, and CID font name used in PDF font aliasing for multibyte fonts.

Table 11–1 CID Font Mapping for PDF Font Aliasing

Language	Oracle NLS_CHARACTERSET Name	CMap Name	CID Font Name
Japanese	JA16SJIS	90ms-RKSJ-H	"KozMinPro-Regular-Acro" (*)
	JA16EUC	EUC-H	"HeiseiKakuGo-W5-Acro" (**)
			"HeiseiMin-W3-Acro" (**)
Korean	KO16KSC5601	KSC-EUC-H	"HYSMyeongJoStd-Medium-Acro" (*)
	K016MSWIN949	KSCms-UHC-H	"HYGothic-Medium-Acro" (**)
			"HYSMyeongJo-Medium-Acro" (**)
Traditional Chinese	ZHT32EUC	CNS-EUC-H	"MSungStd-Light-Acro" (*)
	ZHT16BIG5, ZHT16MSWIN950	ETen-B5-H	"MHei-Medium-Acro" (**)
			"MSung-Light-Acro" (**)
	ZHT16HKSCS	HKscs-B5-H	"MSungStd-Light-Acro" (*)
Simplified Chinese	ZHS16CGB231280	GB-EUC-H	"STSongStd-Light-Acro" (*)
	ZHS16GBK	GBK-EUC-H	"STSong-Light-Acro" (**)

(*) These fonts are available in Adobe Acrobat Reader 5.0 and later.

(**) These fonts are available in Adobe Acrobat Reader 4.0 and later.

It is recommended that you use Version 5.0 CID fonts (*) in order to avoid unexpected font mapping, which results in multibyte characters overlapping. Version 5.0 fonts are compatible with Adobe Acrobat Reader 5.0 and later.

Note: When you deploy reports created with Reports Builder on Windows to a Solaris or Linux Reports Server that has NLS_LANG set to JA16EUC, the PDF font mappings will not work. This happens if any one of the following fonts is used in the PostScript font mapping entry:

"<MS Mincho in JP>....

"<MS PMincho in JP>....

To work around this issue, you must perform one of the following tasks:

- Set *NLS_LANG* to *JA16SJIS*
- Use *MS Gothic* instead of *MS Mincho*. in PostScript font mappings when creating the reports.

11.1.2.1.1 Setup

If font aliasing is necessary, to define the aliasing, instead of directly editing the `uifont.ali` file as in prior releases. For information about font configuration, see Section 6.4.1, "Configuring Fonts".

For more information about the `uifont.ali` file, refer to uifont.ali in Section 9.3, "Font Configuration Files":

11.1.2.1.2 Troubleshooting

If font aliasing does not work, verify that:

- In Acrobat Reader 6.0 and later, choose **File > Document Properties > Fonts.** (In prior releases, beginning with Acrobat Reader 3.0, choose **File > Document Info > 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 3.0 or higher, as required for multibyte character reports to display properly.

11.1.2.2 Font Subsetting

With font subsetting, the PDF file includes the font information needed to *render* the PDF, regardless of the availability of that font on the machine used to view the report. PDF font subsetting works for single byte, multibyte, and Unicode fonts and is the preferred method of creating multibyte reports.

When you subset a font in a PDF file, the font information is embedded into the PDF output for only those characters that are needed for the report output.

Note: You can modify the PDF file if you have:

1. The fonts used in the report installed on your machine.
2. A PDF writer.

11.1.2.2.1 Setup

Before using font subsetting, 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.

> **Note:** The uifont.ali file is located in the following directory:
>
> *$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin*

The section for font subset in the uifont.ali file is [PDF:Subset] and the entry is:

```
[PDF:Subset]
font_name = "font_file_name"
```

where

font_name is the font name, which must be enclosed in quotes if it contains more than one word.

font_file_name is the font file name, which must always be enclosed in quotes, and is case-sensitive. If it does not exactly match the existing font file name, Oracle Reports generates a REP-1924 error.

The font files can be saved in any folder; for example, *DOMAIN_HOME*/reports/*font_folder*. The font file's path is added by default to the $REPORTS_PATH environment variable.

> **Note:** The *font_file_name* is not the font name displayed in Oracle Reports Builder.

Example 1

```
[PDF:Subset]
Arial = "arial.ttf"
```

To use TrueType fonts in a TrueType Collection (.ttc) file, the syntax for the entry in the [PDF:Subset] section in uifont.ali is:

```
[PDF:Subset]
font_name = "ttc_file_name[,table_directory_number]"
```

where

font_name is the font name, which must be enclosed in quotes if it contains more than one word.

ttc_file_name is a TrueType Collection file name.

table_directory_number is the Table Directory number for the TrueType font in a TrueType Collection file, using a zero-based index (for example, "MS PGothic" = "msgothic.ttc,1" indicates that Oracle Reports should use the second font in the

TrueType Collection file). If the *table_directory_number* is omitted or if you supply an invalid value, Oracle Reports will always subset the first font program in the TrueType Collection file.

Example 2

```
[PDF:Subset]
"MS PGothic" = "msgothic.ttc,1"
"MS UI Gothic" = "msgothic.ttc,2"
```

Table 11–2 shows the font name and Table Directory number values for common East Asian TrueType Collection files on the Windows platform.

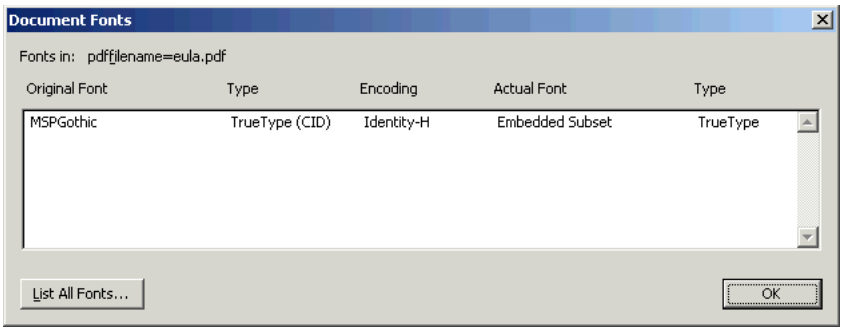
Table 11–2 Common East Asian TrueType Collection Files on the Windows Platform

TTC File Name	Font Name	Table Directory Number
batang.ttc	Batang	0
	BatangChe	1
	Gungsuh	2
	GungsuhChe	3
gulim.ttc	Gulim	0
	GulimChe	1
	Dotum	2
	DotumChe	3
mingliu.ttc	MingLiU	0
	PMingLiU	1
msgothic.ttc	MS Gothic	0
	MS PGothic	1
	MS UI Gothic	2
msmincho.ttc	MS Mincho	0
	MS PMincho	1
simsun.ttc	SimSun	0
	NSimSun	1

You can view the fonts used in your reports as follows:

- In Acrobat Reader 6.0 and later, choose **File > Document Properties > Fonts.** (In prior releases, beginning with Acrobat Reader 3.0, choose **File > Document Info > Fonts**.)
- The Document Font dialog box displays Original Font, Type, Encoding, Actual Font (or the font used), and Type.

Figure 11–2 Font Subsetting



Note: In the case of font subsetting:

- The **Encoding** column will display **Identity-H**.
- The **Actual Font** column will display **Embedded Subset**.
- The **Type** column will display **TrueType**.

11.1.2.2.2 Backward Compatibility

Two environment variables allow for backward compatibility with the font mechanisms of prior releases:

- REPORTS_ENHANCED_FONTHANDLING specifies whether to use the new 12*c* Release (12.2.1.3) font model, or revert to the Toolkit font handling mechanism of 11.1.2/11.1.1. By default, `REPORTS_ENHANCED_FONTHANDLING=YES`.
- REPORTS_ENHANCED_SUBSET specifies whether to include the enhanced TTF font subsetting feature introduced in Oracle Reports 11.1.2/11.1.1. By default, `REPORTS_ENHANCED_SUBSET=YES` to ensure that the PDF file generated is accessible and searchable.

  You can set environment variable `REPORTS_ENHANCED_SUBSET=NO` to revert to the implementation of font subsetting used in releases prior to Oracle Reports 11.1.2/11.1.1; that is, Type3 fonts.

  If you set `REPORTS_ENHANCED_SUBSET=NO`, use Adobe Acrobat Reader and perform the following steps to ensure optimum viewing:

  1. Choose **Edit > Preferences > Page Display**.
  2. Select **Smooth Text, Smooth Line Art, and Smooth Images**.
  3. (Laptop/LCD Screens) Select the **Use CoolType** check box.
  4. Click **OK.**

  Note: These steps are valid for Adobe Acrobat Reader 7.0.

11.1.2.2.3 Troubleshooting If font subsetting does not work, verify the following:

- The fonts you use in the report have bold, italic, and bold italic versions. If you have used italic or bold styles in the report, with PDF font subsetting, and you do not see italic or bold styles in the output, check the Windows TTF files. On Windows, there are some fonts that have bold, italic, and bold italic versions. For example, Arial has `arialbd.ttf` (Arial bold), `ariali.ttf` (Arial italic), and

arialbi.ttf (Arial bold italic), while some other fonts, such as Arial Unicode MS (ARIALUNI.TTF), do not have any bold or italic versions. For fonts that do not have bold or italic versions, Windows synthesizes bold or italic styles from the main font file while displaying, as does Oracle Reports on Windows. These styles are preserved in HTML/HTMLCSS, RTF, and PDF (without PDF subsetting or embedding) outputs. However, while doing the PDF subsetting or embedding, since actual font glyphs are included in the report, Oracle Reports needs the TTF files that contain styles; that is, to include the bold style for Arial in the report, it would need arialbd.ttf. But for fonts such as Arial Unicode MS that do not have such TTF files, PDF subsetted output will not have bold or italic styles.

- The Actual Font value is **Embedded Subset** and Type is **TrueType** (in Acrobat Reader 6.0 and later, choose **File > Document Properties > Fonts**; in prior releases, beginning with Acrobat Reader 3.0, choose **File > Document Info > Fonts**). If this is not specified, then font subsetting is not implemented. 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 case of the font file name matches the case defined in the file.
- The font types are TrueType; that is, *filename*.ttf or *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 3.0 or higher, as required for multibyte character reports to display correctly.
- The value of the REPORTS_ENHANCED_SUBSET environment variable is set to YES. If REPORTS_ENHANCED_SUBSET=NO, Oracle Reports reverts to the earlier implementation of font subsetting, using Type3 fonts to create a PDF document. Type3 fonts are imaged characters that look slightly bolder than they would if expressed as a Type1 font. See Section 11.1.2.2.2, "Backward Compatibility" for more information on improving the viewing quality.

11.1.2.3 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 will work only if the fonts are included in the PDF file. Font embedding increases your PDF file size.

PDF font embedding in Oracle Reports is for Type1 fonts only (single byte 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.

11.1.2.3.1 Setup

The setup for PDF embedding includes:

- A command line keyword: `PDFEMBED`
- A uifont.ali File Entry: `[PDF:Embed]`

PDFEMBED

The command line keyword `PDFEMBED` is used to specify whether Oracle Reports will embed the Type1 PostScript fonts specified in the `uifont.ali` file into the PDF output. See Section A.7.25, "PDFEMBED".

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"

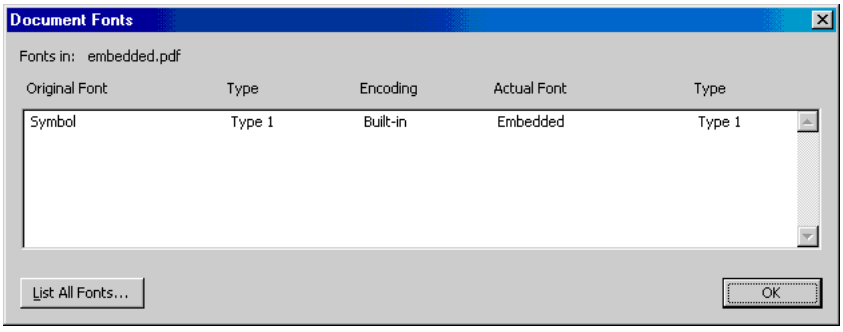
Example 11–1 Font Embedding

```
[PDF:Embed]
Symbol = "Symbol.pfm Symbol.pfb"
```

In Example 11–1, 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 11–3).

Figure 11–3 Font Embedding



11.1.2.3.2 Troubleshooting

If PDF font embedding does not work, verify the following:

- In Acrobat Reader 6.0 and later, choose **File > Document Properties > Fonts.** (In prior releases, beginning with Acrobat Reader 3.0, choose **File > Document Info > 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 *fontfilename*.pfb/.pfa".

  For example (Windows):

  ```
  UtopiaMediumItalic = "UtopiaMediumItalic.pfm UtopiaMediumItalic.pfb"
  ```

- The font name is correct and available on the machine where the PDF file is generated.

11.1.2.4 Font Feature Summary

Table 11–3 summarizes the advantages and disadvantages of font aliasing, font embedding, and font subsetting.

Table 11–3 Comparison of PDF Font Features

PDF Type	Advantages	Disadvantages	PDF Type
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 Aliasing
Font Embedding	Guaranteed display.	Only single byte support provided. Large file size.	Font Embedding
Font Subsetting	Unicode support. Guaranteed display. Generated file is searchable and editable using Adobe Acrobat.	No styles (Italic and Bold) support.	Font Subsetting

11.1.3 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 subsetting takes over font embedding (intermediate).
3. Font embedding takes no precedence (lowest).

For example, if you have included the same font entries for both font embedding and font subsetting, then font subsetting will override font embedding. This is assuming you have not set the command line option `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.. = "arialbd.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"
```

11.1.4 Encryption, Password Protection, and Permissions Security

Beginning with Oracle Reports 12*c* Release (12.2.1.3), you can encrypt and password-protect PDF reports generated by Oracle Reports.

This optional functionality avoids unauthorized reading and changing of PDF reports. The encrypted PDF reports are readable by Acrobat Reader 5.0 and later, and other readers supporting PDF 1.4. This functionality is also compatible with reports developed with prior releases of Oracle Reports.

Oracle Reports uses the Adobe Standard Security Handler to encrypt PDF reports. This standard security handler allows up to two passwords (owner and user) and 8 types of access permissions to be specified for a document.

To provide encryption, password protection, and permissions security in PDF reports, Oracle Reports 12*c* Release (12.2.1.3) introduces the following command line keywords:

- PDFUSER
- PDFOWNER
- PDFSECURITY

> **Note:** To generate PDF encrypted report output using Oracle Reports Builder (`rwbuilder`), you must pass at least one of these command line keywords in the command while starting the `rwbuilder`.
>
> In Oracle Reports Builder, select **Generate to File>PDF** to generate the PDF encrypted output.

Table 11–4 describes the effect of the possible combinations of the `PDFOWNER` and `PDFUSER` command line keywords.

Table 11–4 Effect of PDFUSER and PDFOWNER Keyword Combinations

PDFUSER specified?	**PDFOWNER specified?**	**Effect**
Yes	Yes	When an end user attempts to open PDF report output in Acrobat Reader (5.0 or later), a password prompt displays to request the password specified by `PDFUSER` or `PDFOWNER` to open the document, decrypt it, and display it on the screen. If the end user attempts to change permissions on the PDF report output in Acrobat Writer (6.0 or later), a password prompt displays to request the password specified by `PDFOWNER` to change the document's passwords and permissions.
Yes	No	When an end user attempts to open PDF report output in Acrobat Reader (5.0 or later), a password prompt displays to request the password specified by `PDFUSER` to open the document, decrypt it, and display it on the screen. If the end user attempts to change permissions on the PDF report output in Acrobat Writer (6.0 or later), a password prompt displays to request the same password specified by `PDFUSER` to change the document's passwords and permissions.
No	Yes	When an end user attempts to open PDF report output in Acrobat Reader (5.0 or later), Oracle Reports opens the document, decrypts it, and displays it on the screen. If the end user attempts to change permissions on the PDF report output in Acrobat Writer (6.0 or later), a password prompt displays to request the password specified by `PDFOWNER` to change the document's passwords and permissions.
No	No	Any end user can open PDF report output in Acrobat Reader (5.0 or later), and also change the document's passwords and permissions in Acrobat Writer (6.0 or later). No password prompts display.

For information about suppressing specific permissions for encrypted PDF report using the `PDFSECURITY` command line keyword, see Section A.7.27, "PDFSECURITY".

The encrypted PDF document's passwords and permissions, as specified by PDFUSER, PDFOWNER, and PDFSECURITY, are stored with the PDF document. An end user with authorization to change these values can do so as follows:

1. In Acrobat Writer 6.0 or later, open the PDF document.
2. Select **Document > Security > Restrict Opening and Editing**.
3. In the password prompt, enter the appropriate password (as specified in Table 11–4).dialog box that displays, make desired changes to passwords and permissions.
4. In the dialog box that displays, make desired changes to passwords and permissions.

Support for PDF Security in Distribution

Oracle Reports 12*c* Release (12.2.1.3) supports PDF encryption in distribution and bursting of reports. With this feature, you can set individual passwords and security permissions for each PDF that you generate.

To use this feature, you must add at least one property (pdfuser, pdfowner, and pdfsecurity) in the distribution xml file.

```
<destinations>
 <file id="F1" name="personal_report" format="pdf">
  <property name="pdfsecurity" value="NOCOPY"/>
  <property name="pdfuser" value="mypdf"/>
  <property name="pdfowner" value="employee"/>
   <include src="mainSection"/>
  </file>
</destinations>
```

For more information about the distribution xml file, see Section 20.2, "Introduction to Distribution XML Files" and Section 20.4, "Distribution XML File Examples".

11.1.5 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.

For information on enabling accessibility-related features offered through Oracle Reports from the command line, see Section A.5.1, "ACCESSIBLE". For information about using the Oracle Reports accessibility properties designed to make PDF report output accessible to the disabled community (Alternative Text, Headers, ID, Report Language, and Table Caption properties), see the *Oracle Reports online Help*.

Additionally, refer to Chapter 43, "Building an Accessible JSP-based Web Report" in the *Oracle Reports User's Guide to Building Reports* manual, and to the Oracle accessibility site on OTN (http://www.oracle.com/accessibility/index.html), where you can learn more about accessibility and find the *Creating Accessible Enterprise Reports Using Oracle Reports* white paper.

11.1.6 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 cataloging or searching for documents in external databases.

Oracle Reports provides report-level properties to enable such a classification, known as taxonomy. They are:

- Title
- Author
- Subject
- Keywords

Table 11–5 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

Refer to the *Oracle Reports online Help* for more information on the taxonomy properties.

11.1.7 Graph Support

Oracle Reports provides the capability to specify the dots per inch (DPI) value for the image resolution of the graph in PDF output. This enables you to scale the graph without compromising on the image quality.

See Section B.1.51, "REPORTS_GRAPH_IMAGE_DPI" and Section B.1.54, "REPORTS_JPEG_QUALITY_FACTOR".

11.2 Generating a Unicode PDF File

This section outlines the steps involved in generating a PDF file with a Unicode character set. Before using the font features covered in this section, refer to Table 11–3 to determine which feature best suits your application needs.

11.2.1 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 the following directory on Windows and UNIX:

- *`${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/`*

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`.

11.3 Generating a Bidirectional (BiDi) PDF File

This section outlines the steps involved in generating a PDF file for bidirectional (BiDi) languages. Before using the font features covered in this section, refer to Table 11–3 to determine which feature best suits your application needs.

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, ENHANCED` or `UNICODE`.

   See Also: Section B.1.30, "REPORTS_BIDI_ALGORITHM"

2. `REPORTS_ARABIC_NUMERAL`

   This environment variable specifies the numeric format for Arabic PDF output.

   See Also: Section B.1.28, "REPORTS_ARABIC_NUMERAL"

11.3.1 Font Subsetting

The following example assumes you are using Arabic environment. 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 the following directory on Windows and UNIX:

- *$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin*

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`.

11.4 Generating a Multibyte PDF File

This section outlines the steps involved in generating a PDF file with multibyte fonts. Before using the font features covered in this section, refer to Table 11–3 to determine which feature best suits your application needs.

In PDF font subsetting output, you may see a Wave Dash (`U+301C`) instead of a Fullwidth Tilde (`U+FF5E`). This is due to incompatibility in character mapping between Microsoft and other vendors. To avoid this issue, you can use either `JA16SJISTILDE` or `JA16EUCTILDE` character set for PDF font subsetting. This issue, however, is not observed with the PDF font aliasing feature.

11.4.1 Font Aliasing

Refer to Table 11–1 for a summary of mapping between Oracle `NLS_CHARACTERSET`, CMap name, and its CID font name used in PDF font aliasing for multibyte fonts.

The steps involved in generating a PDF file for multibyte fonts using font aliasing 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 the following directory on Windows and UNIX:

- *$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin*

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 must 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 need not install the Japanese font pack.

11.4.2 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 and edit the [PDF:Subset] section to specify the TrueType font name.

Note: The uifont.ali file is located in the following directory on Windows and UNIX:

- *$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin*

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 specified in the example.
5. Run a report with DESTYPE=FILE DESFORMAT=PDF.

11.5 Generating a Barcode PDF File

This section outlines the steps involved in generating a PDF file with barcode information. Before using the font features covered in this section, refer to Table 11–3 to determine which feature best suits your application needs.

11.5.1 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 the following directory on Windows and UNIX:

- *$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin*

Example

```
[ PDF:Embed ]
SAdHC39a = "SAdHC39a.pfm SAdHC39a.pfb"
```

3. Create a report having Barcode data and set its font to the one specified in the example.
4. Run a report with DESTYPE=FILE DESFORMAT=PDF.

11.5.2 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 and edit the `[PDF:Subset]` section to specify the TrueType font name.

   Note: The `uifont.ali` file is located in the following directory on Windows and UNIX:

   - *$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin*

   Example

   ```
   [ PDF:Subset ]
   SAdHC39a = "SAdHC39a.ttf"
   ```

3. Create a report having barcode data and set it to the font specified in the example.
4. Run a report with `DESTYPE=FILE DESFORMAT=PDF`.

12

Font Model and Cross-Platform Deployment

Modern business needs warrant a seamless integration and interaction across any platform and infrastructure. Oracle Reports enables businesses to develop and deploy information to all levels within and outside of the organization. However, since any enterprise reporting tool is bound to use some platform-specific functionality like the system fonts or printer fonts, there exists a possibility that the look-and-feel of the report changes when the report is ported from one platform to another; for example, from the development platform (commonly Windows) to the deployment platform (commonly a UNIX-based platform).

If the new font model is not used, the user needs to manually configure the font settings as present in pre-11*g* and apply a few fixes manually when deploying your reports on UNIX platforms.

This chapter also covers those scenarios where the choice of platform may affect the look-and-feel of the report output. Each report output format (for example, PDF, HTMLCSS, and RTF) that is open to cross-platform issues is covered in a separate section. These sections provide step-by-step instructions that will ensure that your report output looks the same on all platforms. These guidelines are followed by troubleshooting information and FAQs. Since multibyte and Unicode reports involve some additional steps, separate sections are devoted to those topics.

Before you proceed, it is strongly recommended that you are familiar with the concepts and terminology outlined in the following chapters:

- Chapter 7, "Configuring Oracle Reports Services"
- Chapter 9, "Managing Fonts in Oracle Reports"
- Chapter 10, "Printing on UNIX with Oracle Reports"
- Chapter 11, "Using PDF in Oracle Reports"
- Appendix B, "Environment Variables"

This chapter includes the following sections:

- Generating Unicode PDF Output
- Generating PostScript Output

Note: This chapter lists only those scenarios and guidelines that need additional work to ensure similar outputs across platforms.

12.1 Overview of the Font Model

Oracle Reports uses a new font model that supports the TTF and TTC font types on UNIX platforms. Oracle Reports uses the new font model during runtime, and it uses the old Motif toolkit during design time. The font model applies to all destination formats and supports font aliasing and font subsetting. If the REPORTS_ENHANCED_FONTHANDLING environment variable is set to NO, the old toolkit mechanism is used.

The new font model offers the following features and benefits:

- Simplifies setup, configuration, and discovery of fonts.
- Supports TrueType Fonts (TTF) and TrueType Collections (TTC) on UNIX.
- Supports Unicode font subsetting in PDF on UNIX.
- Includes a simple font lookup algorithm and an enhanced formatter.
- Automatically recognizes new fonts.
- Supports all character sets in PDF.
- Provides font-related diagnostics and tracing information.
- Supports backward-compatibility with the previous font model based on the motif toolkit.
- Uses commonly available TTF and TTC fonts as on Windows.
- Eliminates the need to convert TTF to AFM or TFM files.

Note: It is recommended that you use the Windows version of Oracle Reports Builder to design reports.

12.1.1 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`.

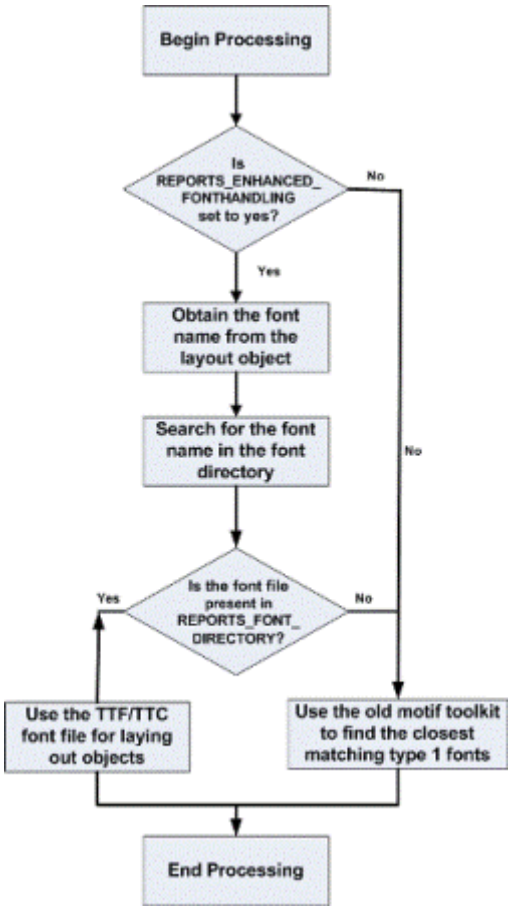
See Also:

- Section 9.3, "Font Configuration Files"
- Section 10.8.1, "ScreenPrinter"

12.1.1.1 Font Lookup Algorithm

illustrates the process of determining the available fonts for generating report output on UNIX.

Figure 12–1 Font Algorithm



The following steps describe how Oracle Reports generates a list of the available fonts for generating output (for example, for the screen, printer, or file):

- Oracle Reports checks whether the `REPORTS_ENHANCED_FONTHANDLING` environment variable is set.
- If the variable is set, it checks whether the TTF and TTC fonts used in the layout objects are present in REPORTS_FONT_DIRECTORY.
- If the TTF and TTC fonts are present in `REPORTS_FONT_DIRECTORY`, Oracle Reports uses the TTF font in calculating metrics, which prevents misalignment in the multibyte language report output.
- If the TTF and TTC fonts are not in the `REPORTS_FONT_DIRECTORY`, Oracle Reports reverts to the old toolkit mechanism, which finds the nearest matching Type 1 font on the machine.

Note: If the REPORTS_ENHANCED_FONTHANDLING variable is not set, the old motif toolkit mechanism is used.

12.1.2 Configuring the New Font Model

To configure the new font model, complete the following steps:

1. Ensure that the REPORTS_ENHANCED_FONTHANDLING environment variable is set to yes. The default value is yes.
2. Copy all the TTF and TTC files, which are used in the report, to the REPORTS_FONT_DIRECTORY. The default font directory is $DOMAIN_HOME/reports/fonts.
3. Remove any unnecessary aliasing from the uifont.ali file. For example, Arial is aliased to Helvetica, by default. If your report uses the Arial font, you must remove the aliasing from the uifont.ali file.

Note: If you choose to use the old motif toolkit, you must consider cross-platform issues and apply workaround solutions. It is recommended that you use the new font model for developing and deploying reports on UNIX.

12.1.3 Font Diagnosis and Tracing

The new font model in Oracle Reports 12*c* Release (12.2.1.3) provides improved font diagnosability and tracing.

You can configure log levels for persistent loggers and active runtime loggers. Log levels allow you to limit the amount of tracing information included in your tracing output.

For example, if you set **Oracle Diagnostic Logging Level (Java Level) = Trace:1 (FINE)**, the following font-related information is included in your log files and tracing output:

- Name of the font directory
- Fonts available in the font directory
- Fonts used by objects in a paper layout that are not available in the font directory

For more information about diagnosing font issues, see Section 9.8, "Diagnosing Font Issues".

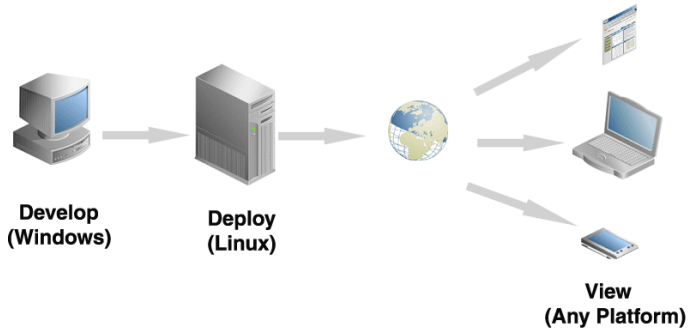
12.2 Overview of Cross-Platform Issues

Oracle Reports is available on many platforms, including Windows, Linux, Sun Solaris, HP-UX, and IBM AIX. You can use Oracle Reports to develop and deploy reports on any of these platforms interchangeably. The most common scenario is that the report is developed on Windows, and is deployed on a UNIX-based environment, such as Linux (see Figure 12–2). This may result in a slight change in the look-and-feel of the deployed report. For example, when you are developing the report on Windows, you allocate enough space to each text object or field in your report. However, when you deploy and run the report on Linux, you may see that the text does not fit within the allocated space in the output. Such issues that are the direct result of change in platform are referred to as cross-platform issues. A possible cause of such issues is that the fonts available on the development platform are not available on the deployment

platform. As a result, when the report is executed on the deployment platform, a substitute font needs to be used for formatting the report output. Since any two fonts are likely to have certain differences, the report output on the development and deployment platforms looks different.

Another likely scenario in which you may encounter cross-platform issues is when the platform on which the report is finally viewed (see Figure 12–2) does not have the proper fonts installed. Thus, even if the development and deployment platforms display the report output correctly, the platform on which the end-user views the report will not display the proper look-and-feel of the report.

Figure 12–2 Sample Cross-Platform Deployment Scenario



12.2.1 Font Availability On Different Platforms

A font is a set of printable or displayable text characters in a specific style and size. Fonts are needed for displaying the report on the screen as well as for printing it. The metrics for these fonts are picked up by Oracle Reports while formatting the report; that is, while executing the report command. Based on the font metrics, the report is formatted and the output is produced.

The font metrics are provided by specific files that must be available on the system where you are running Oracle Reports Services. On Windows, these font metrics are provided by True Type Font (TTF) files or True Type Collection (TTC) files. On UNIX platforms, the font metrics are taken from Adobe Font Metrics (AFM) files or TeX Font Metrics (TFM) files. The font availability and the metrics can vary based on the operating system used. This difference in fonts used and the rendering can affect the visual appearance of the generated output.

Example 1: Tahoma, a commonly used font in single-byte regions, is available on Windows but not on UNIX. For example, a reports developer has used Tahoma font while designing the report. The output of the report looks good on the development platform; that is, Windows. The report is then ported to the deployment platform (say Linux). When you submit a request to the Reports Server to execute this report, the Reports Server looks for Tahoma font metrics. It will be unable to find the metric file, since Tahoma is a Windows-specific font. Another font that closely resembles Tahoma will be used instead. This will affect the report output since a different font has been used.

Example 2: The development as well as deployment platform is Windows. So Reports Servers on both the development and deployment platforms are able to access Tahoma font since both run on Windows. However, suppose an end-user views the output on Linux. All reports output formats (HTML, HTMLCSS, RTF, and PDF) merely refer to the fonts and do not embed the fonts in the output unless you specifically use the font embedding feature in PDF. As a result, the client system will look for the Tahoma font

to display the report output on client machine. Since Tahoma is not available on Linux, the user will encounter cross-platform issues while viewing the output.

12.2.2 Fixing Font-Related Issues

As we have seen, many cross-platform issues are caused by the non-availability of fonts either on the production environment (where the Reports Server is running) or on the client system. These font availability issues must be resolved by a 3-step approach:

1. **Development platform**: Ensure that you develop the report keeping in mind the font availability on the deployment platform. All font files that are available on Windows (TTF files) may not be available on UNIX (AFM or TFM files). If you have the correct AFM or TFM font file available on the UNIX platform, you can continue to use it (AFM for PostScript printing and TFM for PCL). For fonts with AFM files not readily available on UNIX, or if you encounter any font issues in the report output such as text misalignment, you can convert and generate an AFM file from the Windows TTF file using freely available third party utilities, such as `ttf2pt1`. Do not attempt to convert to a TFM file, as this may not produce reliable results.
2. **Deployment platform**: Ensure that the fonts used in the report are available. For PDF output, use font aliasing to substitute the unavailable font with the closest available font. Global font aliasing can be used for all output formats.
3. Same comment holds for all the subsequent sections where you have asked to provide examples.
4. **Client platform**: Ensure that you account for font unavailability on the client system. For example, in the case of PDF output, you can use Font Subsetting or Font Embedding, as described in Chapter 11, "Using PDF in Oracle Reports". In the case of HTML, HTMLCSS or RTF output formats it is not possible to embed the fonts, so it is best to design the report using fonts that are known to be available on all platforms.

12.3 Generating HTMLCSS, RTF, or Web Output

Table 12–1 shows the cross platform deployment scenario where the destination format is HTMLCSS, RTF, or the Web.

Table 12–1 Cross Platform Deployment - Scenario 1

Development Platform	Deployment Platform	Destination Format
Windows	UNIX	HTMLCSS, RTF, or Web

This section discusses designing and deploying a report for HTMLCSS, RTF, or Web output in the following subsections:

- Designing Your Report
- Deploying Your Report
- Frequently Asked Questions

12.3.1 Designing Your Report

To prepare your report before you deploy it on a UNIX platform:

1. Create a new report. While creating your report ensure that you leave additional padding space for boilerplate and field objects. This is to ensure that the box's size accounts for any possible increase in the text width when the report is run on the deployment platform.
2. Use only those fonts in your report that:
   - Are available on UNIX. All font files that are available on Windows (TTF files) may not be available on UNIX (AFM or TFM files). If you have the correct AFM or TFM font file available on the UNIX platform, you can continue to use it (AFM for PostScript and TFM for PCL).

     Note: AFM support is extended only to single-byte PostScript file generation, with the exception of Japanese encoding.

     The encoding schemes supported for the AFM files are:

     `AdobeStandardEncoding`

     `ExtJIS12-88-CFEncoding`

     `FontSpecific`

     `HRoman`

     `ISOLatinHebrew`

     `JIS12-88-CFEncoding`

     `JIS12e-88-CFEncoding`

   - Can scale well. For example, MS Sans Serif does not scale well to a different size, whereas Tahoma does. The reason is that the MS Sans Serif font is a raster font that does not scale well to any size and usually has rounding issues. On the other hand, Tahoma font is a TrueType font that is very similar in visual appearance to the MS Sans Serif font. Additionally, Tahoma is a vector font that can be scaled to any size and rotated to any angle.

12.3.2 Deploying Your Report

Deploying a Report in 11*g* that uses the New Font Model

1. Ensure that the `REPORTS_ENHANCED_FONTHANDLING` environment variable is set to yes. The default value is yes.
2. Copy all the TTF and TTC files, which are used in the report, to the `REPORTS_FONT_DIRECTORY`. The default font directory is `$DOMAIN_HOME/reports/fonts`.
3. Remove any unnecessary aliasing from the `uifont.ali` file. For example, Arial is aliased to Helvetica, by default. If your report uses the Arial font, you must remove the aliasing from the uifont.ali file.
4. Run the Report.

Deploying Reports in Pre-11*g* Version that uses Motif Tool Kit Mechanism

For fonts with AFM files not readily available on UNIX, or if you encounter any font issues in the report output such as text misalignment, you can convert and generate an AFM file from the Windows TTF file using freely available third party utilities, such as `ttf2pt1`. Do not attempt to convert to a TFM file, as this may not produce reliable results.

To deploy your report on a UNIX platform when AFM font files are not available:

1. Locate the TTF files corresponding to the fonts used in your report. Convert these TTF files to AFM to ensure that you will have the AFM files for the fonts used in your report.

   Use a True Type to Type 1 font converter utility to convert the TTF files to AFM files. For example, `ttf2pt1`.

2. Post-conversion, remove the `.afm` extension in the AFM file name. For example:

Table 12–2 Post Conversion Font File Names

Before Converting	After Converting	After Renaming
`arial.ttf`	`arial.afm`	`Arial`
`cour.ttf`	`cour.afm`	`CourierNew`

3. Copy the converted AFM files to the `$ORACLE_HOME/guicommon/tk/admin/AFM` directory.

4. Edit the `screenprinter.ppd` file with any text editor.

   Note: If you have defined a default printer by including an entry in *ORACLE_HOME*`/guicommon/tk/admin/uiprint.txt`, you must add the appropriate entries in the printer's PPD file for a PostScript printer or in the HPD file for a PCL printer.

   If you have not set up a default printer:

   - A default printer surface that mimics the screen (`screenprinter.ppd`) is used for formatting.
   - You must add the necessary font and resolution entries in the `screenprinter.ppd` file.

   The PPD and HPD files are located at:

   - `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/PPD`
   - `$ORACLE_HOME/guicommon/tk/admin`

   Oracle Reports searches for HPD or PPD files initially in the Oracle Instance location and then in the Oracle Home location.

   Refer to Section 10.8.1, "ScreenPrinter" for more information on the `screenprinter.ppd` file.

   Ensure that the PPD/HPD file used contains an entry for each AFM or TFM file that you use in your report. PPD/HPD files are configuration files containing printer driver settings and the list of all the fonts supported by the printer.

   Navigate to the to the `Font Information` section in the PPD file and add the necessary entries for the font files in the following format:

```
*FONTNAME:ENCODING:VERSION:LOCATION
```

   For example:

```
*Font Arial: Standard "(Version 2.76)" Standard ROM
*Font CourierNew: Standard "(Version 2.76)" Standard ROM
```

Ensure that the AFM file name exactly matches the font name specified in the PPD file as Oracle Reports searches for this file based on the font name in the PPD file.

5. Ensure that the fonts used in your report are not aliased.

   For example, edit the `uifont.ali` file and comment the entries in the `[Global]` section, where Arial and Courier New are aliased to Helvetica and Courier, respectively.

   ```
   [ Global ] # Put mappings for all surfaces here.
   # Mapping from MS Windows
   #Arial = helvetica
   #"Courier New" = courier
   ```

   This ensures that Arial and Courier New are not aliased to any other font.

   Note: The `uifont.ali` file is located in the following directory on Windows and UNIX:

   - On windows: *$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\common*
   - On UNIX: `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin`

   Use font aliasing only if you are unable to generate the AFM file for a particular font. You can then alias the missing font to the closest match. The fonts must be made available on the machine displaying the report output and not necessarily on the machine generating the report output.

   ---

6. Run the report.

   ```
   http://mywebserver.com:reports/rwservlet?server=myserver+report="/home/myreports/test.rdf"+authid=hr/hr@mydb+desformat=htmlcss+destype=cache
   ```

   The HTMLCSS output of your report will look exactly the same as the one generated on Windows.

12.3.2.1 Troubleshooting Information

If you encounter deployment issues, review the following troubleshooting information:

- If you do not get the correct fonts in the HTMLCSS output, set the environment variable `DEBUG_SLFIND` to a log file name, for example, `debug.txt`, and run the report. The font files that are looked up while parsing the PPD file as well as the fonts used will be written to the log file `debug.txt`. Specifically, check for the following:

  1. The PPD file that you modified should be picked up. If it is not picked up it is a configuration issue. Refer to Chapter 9, "Managing Fonts in Oracle Reports".
  2. The AFM files that you have copied to AFM directory should be picked up next.

  See Chapter 10, "Printing on UNIX with Oracle Reports" for more information on `DEBUG_SLFIND`.

12.3.3 Frequently Asked Questions

This section covers frequently asked questions (FAQs) pertaining to deploying a report to HTMLCSS, RTF or the Web.

Question

When I design a report on Windows with font styles such as italic and bold, then run the report on UNIX, I do not see the output as it appeared on Windows. Why?

Answer

On UNIX, report formatting is done using fonts' corresponding AFM files. By default, these AFM files are picked from the `$DOMAIN_HOME/guicommon/tk/admin/AFM` directory, provided as part of the installation. If the font style used in report does not have a corresponding AFM file on UNIX, the closest matching AFM file is used. For example, if you design a report on Windows with Courier Italic font, then run the report on UNIX, you may see only plain Courier font in the output. This happens because there is no AFM file available for Courier Italic font in the `$ORACLE_HOME/guicommon/tk/admin/AFM` directory, so instead Courier is picked. To work around this issue, you can alias your font's style to the same style for some other font that has AFM available. For example, you could alias Courier Italic to Times Italic in the `global` section of `uifont.ali`. Moreover, on Windows, there are some fonts that have bold, italic, and bold italic versions; for example, Arial has `arialbd.ttf` (Arial bold), `ariali.ttf` (Arial italic), and `arialbi.ttf` (Arial bold italic). Therefore, if you are using any font that has bold, italic, and bold italic TTF files available, you can generate AFM files from these files using `ttf2pt1` and use these AFM files on UNIX.

Question

My report was created in Windows and is deployed on HP-UX 11. Although the font style on HP-UX 11 is correct, the spacing between the lines is inconsistent and some text is unable to fit in the allocated space. How can I fix the spacing so that my text fits correctly?

Answer

Ensure that you have set up the corresponding AFM files for all the fonts used in your document. Refer to Section 12.3.1, "Designing Your Report" for more information.

Question

My report is designed on Windows. When it is deployed on a different platform, it displays garbled output. For example, some fields display, `*****` instead of the actual content. Is this a spacing issue?

Answer

Oracle Reports cannot find the AFM files of the font that you have used in your report. You can verify this by opening the report's HTML source and searching for the font that you have used.

Oracle Reports then uses the closest matching font whose metrics are bigger than the original font. Therefore, when the characters cannot fit in the box, a `****` is displayed in the field, instead of the actual output.

Ensure that:

- You have set up the AFM files for all the fonts used in your report. Refer to Section 12.3.1, "Designing Your Report" for more information.

- You have left approximately 10% extra padding space for fields and text boilerplates.

Question

When my report is run on UNIX, the HTML or the HTMLCSS output looks shrunk. However, the same report run on Windows looks fine. What can I do to ensure that my report looks the same on UNIX as it did on Windows?

Answer

If you see shrinkage or expansion in your HTMLCSS output and you are on Oracle9*i* Reports, then set the environment variable `REPORTS_DEFAULT_PIXEL_SIZE` to any value ranging from 72 through 200 in `reports.sh` and restart the Reports Server.

For example:

```
REPORTS_DEFAULT_PIXEL_SIZE =72
export REPORTS_DEFAULT_PIXEL_SIZE
```

There will not be any `HTMLCSS` output shrinkage/expansion in Oracle Reports 10*g* as a fixed resolution is picked up from `screenprinter.ppd`. This resolution is editable.

```
*DefaultResolution: 96dpi (recommended)
```

Question

Can I use the `overstrike` property when I deploy a report in Solaris?

Answer

A limitation of AFM files is that it does not support the `overstrike` property.

12.4 Generating Single-Byte PDF Output

Table 12–3 shows the cross-platform deployment scenario where the destination format is single-byte PDF created using PDF font subsetting. For more information on PDF font features, refer to Chapter 11, "Using PDF in Oracle Reports".

Table 12–3 Cross Platform Deployment - Scenario 2

Development Platform	Deployment Platform	Destination Format
Windows	UNIX	PDF (single byte)

This section discusses designing and deploying a report for single-byte PDF output in the following subsections:

- Designing Your Report
- Deploying Your Report in Pre-11g Version That Uses Motif Tool Kit Mechanism
- Frequently Asked Questions

12.4.1 Designing Your Report

To prepare your report before you deploy it on a UNIX platform:

1. Create a new report.
2. Use only those fonts in your report that:

- Are available on UNIX. All font files that are available on Windows (TTF files) may not be available on UNIX (AFM or TFM files). If you have the correct AFM or TFM font file available on the UNIX platform, you can continue to use it (AFM for PostScript and TFM for PCL).

  Note: AFM support is extended only to single-byte PostScript file generation, with the exception of Japanese encoding.

  The encoding schemes supported for the AFM files are:

  ```
  AdobeStandardEncoding
  ExtJIS12-88-CFEncoding
  FontSpecific
  HRoman
  ISOLatinHebrew
  JIS12-88-CFEncoding
  JIS12e-88-CFEncoding
  ```

- Can scale well. For example, MS Sans Serif does not scale well to a different size, whereas Tahoma does. The reason is that the MS Sans Serif font is a raster font that does not scale well to any size and usually has rounding issues. On the other hand, Tahoma font is a TrueType font that is very similar in visual appearance to the MS Sans Serif font. Additionally, Tahoma is a vector font that can be scaled to any size and rotated to any angle.

12.4.2 Deploying Your Report in Pre-11*g* Version That Uses Motif Tool Kit Mechanism

For fonts with AFM files not readily available on UNIX, or if you encounter any font issues in the report output such as text misalignment, you can convert and generate an AFM file from the Windows TTF file using freely available third party utilities, such as `ttf2pt1`. Do not attempt to convert to a TFM file, as this may not produce reliable results.

To deploy your report on a UNIX platform using PDF font subsetting:

1. Locate the TTF files corresponding to the fonts used in your report. Convert these TTF files to AFM to ensure that you will have the AFM files for the fonts used in your report.

   Use a True Type to Type 1 font converter utility to convert the TTF files to AFM files. For example, `ttf2pt1`.

2. Post-conversion, remove the `.afm` extension in the AFM file name. For example:

Table 12–4 Post Conversion Font File Names

Before Converting	After Converting	After Renaming
`arial.ttf`	`arial.afm`	`Arial`
`cour.ttf`	`cour.afm`	`CourierNew`

3. Copy the Windows TTF files that you have used in your report to the fonts directory on your UNIX machine. For example, `$ORACLE_HOME/reports/fonts`.
4. Add the path to the TTF files in the `REPORTS_PATH` environment variable. This ensures that the font files can be referenced by Reports Runtime.

5. Edit the `screenprinter.ppd` file with any text editor.

   Note: If you have defined a default printer by including an entry in *ORACLE_HOME*`/guicommon/tk/admin/uiprint.txt`, you must add the appropriate entries in the printer's PPD file for a PostScript printer or in the HPD file for a PCL printer.

   If you have not set up a default printer:

   - A default printer surface that mimics the screen (`screenprinter.ppd`) is used for formatting.
   - You must add the necessary font and resolution entries in the `screenprinter.ppd` file.

   The PPD and HPD files are located at:

   - `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin`
   - `$ORACLE_HOME/guicommon/tk/admin`

   Oracle Reports searches for HPD or PPD files initially in the Oracle Instance location and then in the Oracle Home location.

   Refer to Section 10.8.1, "ScreenPrinter" for more information on the `screenprinter.ppd` file.

   Ensure that the PPD/HPD file used contains an entry for each AFM or TFM file that you use in your report. PPD/HPD files are configuration files containing printer driver settings and the list of all the fonts supported by the printer.

   Navigate to the to the `Font Information` section in the PPD file and add the necessary entries for the font files in the following format:

   ```
   *FONTNAME:ENCODING:VERSION:LOCATION
   ```

   For example:

   ```
   *Font Arial: Standard "(Version 2.76)" Standard ROM
   *Font CourierNew: Standard "(Version 2.76)" Standard ROM
   ```

   Ensure that the AFM file name exactly matches the font name specified in the PPD file as Oracle Reports searches for this file based on the font name in the PPD file.

6. Copy the converted AFM files to the `$ORACLE_HOME/guicommon/tk/admin/AFM` directory.

7. Ensure that the `uiprint.txt` has the entry for the appropriate PPD file:

   ```
   printer name:PostScript:2:test:ppd file
   ```

   For example:

   ```
   printer1:PostScript:2:test:hpljet42.ppd
   ```

8. Edit the `hpljet42.ppd` file with any text editor.

Note: Copy the PPD file from `ORALCE_HOME/guicommon/tk/admin/PPD` to the following location:

```
${DOMAIN_
HOME}/config/fmwconfig/components/ReportsToolsComponent/<rep
orts_tools_name>/guicommon/tk/admin/PPD
```

---

9. Ensure that the PPD/HPD file used contains an entry for each AFM or TFM file that you use in your report.

   Navigate to the `Font Information` section and add the necessary entries for the new AFM files in the following format:

   ```
   *FONTNAME:ENCODING:VERSION:LOCATION
   ```

   For example:

   ```
   *Font Arial: Standard "(Version 2.76)" Standard ROM
   *Font CourierNew: Standard "(Version 2.76)" Standard ROM
   ```

   Ensure that the AFM file name is the same as the font name given in the PPD file. Oracle Reports searches for this file based on the font name in the PPD file.

10. Ensure that the fonts used in your report are not aliased.

    For example, edit the `uifont.ali` and comment the entries in the `[Global]` section where Arial and Courier New are aliased, by default, to Helvetica and Courier respectively.

    ```
    [ Global ] # Put mappings for all surfaces here.
    # Mapping from MS Windows
    #Arial = helvetica
    #"Courier New" = courier
    ```

    Note: The `uifont.ali` file is located in the following directory on Windows and UNIX:

    - On windows: *$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\common*
    - On UNIX: `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin`

    Use font aliasing only if you are unable to generate the AFM file for a particular font. You can then alias the missing font to the closest match. The fonts must be made available on the machine displaying the report output and not necessarily on the machine generating the report output.

    ---

11. Add the appropriate entries in the `[PDF:Subset]` section to subset the fonts used in your report.

    For example:

    ```
    [ PDF:Subset ]
    "arial" = "arial.ttf"
    "courier new" = "cour.ttf"
    ```

For PDF file portability, you can use either font subsetting or font embedding. File portability ensures that the PDF report does not depend on the machine where it is viewed to have the fonts installed.

12. Run the report to PDF and view it. The PDF should contain the fonts used in your report. For example, Arial and Courier New fonts.

    To verify the fonts used, do the following:

    - In Acrobat Reader 6.0, choose **File** >**Document Properties** > **Fonts**.
    - In Acrobat Reader 3.0 and above, choose **File** >**Document Info** > **Fonts**.

    The Original Font column displays the Arial and Tahoma fonts. The PDF document should not contain any font alignment issues.

12.4.2.1 Troubleshooting Information

If you encounter deployment issues, review the following troubleshooting information:

- If you do not get the correct fonts in the PDF output, set the environment variable `DEBUG_SLFIND` to a log file name (for example, `debug.txt`) and run the report. The font files that are looked up while parsing the PPD file as well as the fonts used will be written to the specified file. Specifically, check for the following:

  1. The PPD file that you modified should be picked up. If it is not picked up it is a configuration issue. Refer to Chapter 9, "Managing Fonts in Oracle Reports".
  2. The AFM files that you have copied to AFM directory should be picked up next.

  See Chapter 10, "Printing on UNIX with Oracle Reports" for more information on `DEBUG_SLFIND`.

12.4.3 Frequently Asked Questions

This section contains frequently asked questions (FAQs) pertaining to deploying a report to single-byte PDF output.

Question

My PDF report page count varies when it is deployed in Windows and UNIX platforms. What must I do to fix it?

Answer

Your report uses the default printer for formatting. Ensure that the same resolution and the same fonts used are made available to both the printers. One way of achieving this would be to generate AFM files from Windows TTF font files and then copy the Windows TTF files and AFM files to UNIX in the appropriate folders. Also set the same resolution as Windows in PPD/HPD files. Follow the process specified in the prior steps.

Question

The page count of my report varies when run on different installations of UNIX. How can I ensure that the page count of my report is the same regardless of the installation?

Answer

In UNIX, Oracle Reports uses the PPD/HPD file of the default printer in the installation for formatting. The resolution and list of fonts will be picked up from this PPD/HPD files. If there is no default printer setup in the installation, then `screenprinter.ppd` will be used. This PPD file emulates the screen. Earlier versions of Oracle Reports used the `DISPLAY` environment variable instead. Ensure that the two installations **use the same AFM/TFM files and font files**, so that the number of pages of PDF output will be the same.

12.5 Generating Multibyte PDF Output

Table 12–5 shows the cross platform deployment scenario where the destination format is multibyte PDF created using PDF font subsetting. For more information on PDF font features, refer to Chapter 11, "Using PDF in Oracle Reports".

Table 12–5 Cross Platform Deployment - Scenario 4

Development Platform	Deployment Platform	Destination Format
Windows	UNIX	PDF (multibyte)

This section discusses designing and deploying a report for multibyte PDF output in the following subsections:

- Designing Your Report in Pre-11g Version That Uses Motif Tool Kit Mechanism
- Deploying Your Report in Pre-11g Version That Uses Motif Tool Kit Mechanism
- Frequently Asked Questions

12.5.1 Designing Your Report in Pre-11*g* Version That Uses Motif Tool Kit Mechanism

To prepare your report before you deploy it on a UNIX platform:

1. Create a new report with a TrueType multibyte font. For example, Simplified Arabic font with the `AR8ISO8859P6` character set.
2. Use only those fonts in your report that:
   - Are available on UNIX. All font files that are available on Windows (TTF files) may not be available on UNIX (AFM or TFM files). If you have the correct AFM or TFM font file available on the UNIX platform, you can continue to use it (AFM for PostScript and TFM for PCL).

Note: AFM support is extended only to single-byte PostScript file generation, with the exception of Japanese encoding.

The encoding schemes supported for the AFM files are:

```
AdobeStandardEncoding
ExtJIS12-88-CFEncoding
FontSpecific
HRoman
ISOLatinHebrew
JIS12-88-CFEncoding
JIS12e-88-CFEncoding
```

- Can scale well. For example, MS Sans Serif does not scale well to a different size, whereas Tahoma does. The reason is that the MS Sans Serif font is a raster font that does not scale well to any size and usually has rounding issues. On the other hand, Tahoma font is a TrueType font that is very similar in visual appearance to the MS Sans Serif font. Additionally, Tahoma is a vector font that can be scaled to any size and rotated to any angle.

12.5.2 Deploying Your Report in Pre-11*g* Version That Uses Motif Tool Kit Mechanism

For fonts with AFM files not readily available on UNIX, or if you encounter any font issues in the report output such as text misalignment, you can convert and generate an AFM file from the Windows TTF file using freely available third party utilities, such as `ttf2pt1`. Do not attempt to convert to a TFM file, as this may not produce reliable results.

To deploy your report on a UNIX platform:

1. Locate the TTF files corresponding to the fonts used in your report. Convert these TTF files to AFM to ensure that you will have the AFM files for the fonts used in your report.

   Use a True Type to Type 1 font converter utility to convert the TTF files to AFM files. For example, `ttf2pt1`.

2. Post-conversion, remove the `.afm` extension in the AFM file name. For example:

Table 12–6 Post Conversion Font File Names

Before Converting	After Converting	After Renaming
`simpo.ttf`	`simpo.afm`	`SimplifiedArabic`

3. Copy the Windows TTF file used in your report to the fonts directory on your UNIX machine. For example, `$ORACLE_HOME/fonts`.
4. Add the path to the TTF file in the `REPORTS_PATH` environment variable.
5. Copy the AFM file to the `$ORACLE_HOME/guicommon/tk/admin/AFM` directory.
6. Edit the `screenprinter.ppd` file with any text editor.

Note: If you have defined a default printer by including an entry in *ORACLE_HOME*/guicommon/tk/admin/uiprint.txt, you must add the appropriate entries in the printer's PPD file for a PostScript printer or in the HPD file for a PCL printer.

If you have not set up a default printer:

- A default printer surface that mimics the screen (screenprinter.ppd) is used for formatting.
- You must add the necessary font and resolution entries in the screenprinter.ppd file.

The PPD and HPD files are located at:

- ${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/
- $ORACLE_HOME/guicommon/tk/admin

Oracle Reports searches for HPD or PPD files initially in the Oracle Instance location and then in the Oracle Home location.

Refer to Section 10.8.1, "ScreenPrinter" for more information on the screenprinter.ppd file.

Ensure that the PPD/HPD file used contains an entry for each AFM or TFM file that you use in your report. PPD/HPD files are configuration files containing printer driver settings and the list of all the fonts supported by the printer.

Navigate to the to the Font Information section in the PPD file and add the necessary entries for the font files in the following format:

```
*FONTNAME:ENCODING:VERSION:LOCATION
```

For example:

```
*Font Arial: Standard "(Version 2.76)" Standard ROM
*Font CourierNew: Standard "(Version 2.76)" Standard ROM
```

Ensure that the AFM file name exactly matches the font name specified in the PPD file as Oracle Reports searches for this file based on the font name in the PPD file.

7. Ensure that the uiprint.txt has the entry for the appropriate PPD file: printername: PostScript:2:test:ppd_file.

   For example:

   ```
   printer1:PostScript:2:test:hpljet42.ppd
   ```

8. Edit the hpljet42.ppd file with any text editor.

Note: Copy the PPD file from ORACLE_HOME/guicommon/tk/admin/PPD to the following location:

$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/PPD

---

9. Ensure that the PPD/HPD file used contains an entry for each AFM or TFM file that you use in your report.

   Navigate to the `Font Information` section and add the necessary entries for the new AFM files in the following order:

   ```
   *FONTNAME:ENCODING:VERSION:LOCATION
   ```

   For example:

   ```
   *Font SimplifiedArabic: Standard "(001.01)" Standard ROM
   ```

10. Ensure that the fonts used in your report are not aliased.

    Edit `uifont.ali` and comment the entries, if any, where Simplified Arabic font is aliased to some other font. For example: `"SimplifiedArabic"="Arial"`.

    Note: The `uifont.ali` file is located in the following directory on Windows and UNIX:

    - Windows: `$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\common`
    - UNIX: *`$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin`*

    Use font aliasing only if you are unable to generate the AFM file for a particular font. You can then alias the missing font to the closest match. The fonts must be made available on the machine displaying the report output and not necessarily on the machine generating the report output.

    ---

11. In the `[PDF:Subset]` section, add the appropriate entries to subset the fonts.

    For example:

    ```
    [ PDF:Subset ]
     "SimplifiedArabic" = "simpo.ttf"
    ```

    For PDF file portability, you can use either font subsetting or font embedding. File portability ensures that the PDF report does not depend on the machine where it is viewed to have the fonts installed.

12. Run the report to PDF and view it.

    ```
    http://mywebserver.com:reports/rwservlet?server=myserver+report=
    "/home/myreports/test.rdf"+authid=hr/hr@mydb+desformat=PDF+destype=cache
    ```

    The PDF should contain the font that you have used in your report. For example, the David font.

    To verify the fonts used, do the following:

    - In Acrobat Reader 6.0, choose **File >Document Properties > Fonts**.
    - In Acrobat Reader 3.0 and above, choose **File >Document Info > Fonts**.

    The Original Font column should display the David font.

12.5.2.1 Troubleshooting Information

If you encounter deployment issues, review the following troubleshooting information:

- If you do not get the correct fonts in the PDF output, set the environment variable `DEBUG_SLFIND` to a log file name, for example, `debug.txt`, and run the report. The font files that are looked up while parsing the PPD file as well as the fonts used will be written to the log file `debug.txt`. Specifically, check for the following:

  1. The PPD file that you modified should be picked up. If it is not picked up it is a configuration issue. Refer to Chapter 9, "Managing Fonts in Oracle Reports".
  2. The AFM files that you have copied to AFM directory should be picked up next.

  See Chapter 11, "Using PDF in Oracle Reports" for more information on `DEBUG_SLFIND`.

12.5.3 Frequently Asked Questions

This section contains frequently asked questions (FAQs) pertaining to deploying a report to multibyte PDF output.

Question

What are the cross-platform issues for the PDF output when using CID multibyte fonts?

Answer

To enable multibyte language support in PDF reports with CID multibyte fonts, you must make sure that the Asian font package is installed for your Acrobat Reader on the machine where you are going to view these PDF files. The Asian font package is available at the Adobe Web site.

Question

Why is the formatting of my report not correct when using multibyte fonts on UNIX, or why do I see misaligned text in the report?

Answer

If you have used `ttf2pt1` to create the AFM file, `ttf2pt1` has a limitation in that it creates AFM files with metrics information for only the first 256 characters of the font. The first 256 characters are for Latin-1 characters. So, if your font contains more than 256 characters, metrics information will not be available for the additional characters. Oracle Reports uses default metrics information contained in the AFM file when it encounters characters in the report that are not in the AFM file. These default metrics may not match the exact metrics of the characters used in the report. For this reason, formatting will not be correct. To avoid this situation when you are using characters beyond the first 256 characters of the font, you can use fixed width fonts where all the characters have the same width. For example, Miriam Fixed is a fixed width font for Hebrew and can be used to avoid formatting issues.

12.6 Generating Unicode PDF Output

Table 12–7 shows the cross-platform deployment scenario where the destination format is Unicode PDF created using PDF font subsetting. For more information on PDF font features, refer to Chapter 11, "Using PDF in Oracle Reports".

Table 12–7 Cross Platform Deployment - Scenario 5

Development Platform	Deployment Platform	Destination Format
Windows	UNIX	PDF (Unicode)

This section discusses designing and deploying a report for Unicode PDF output in the following subsections:

- Designing Your Report in Pre-11g Version That Uses Motif Tool Kit Mechanism
- Deploying Your Report in Pre-11g Version That Uses Motif Tool Kit Mechanism
- Frequently Asked Questions

12.6.1 Designing Your Report in Pre-11*g* Version That Uses Motif Tool Kit Mechanism

To prepare your report before you deploy it on a UNIX platform:

1. Create a new report using a Unicode font. For example, Arial Unicode MS font.
2. Use only those fonts in your report that:
   - Cover the entire Unicode range that your report uses.
   - Are available on UNIX. All font files that are available on Windows (TTF files) may not be available on UNIX (AFM or TFM files). If you have the correct AFM or TFM font file available on the UNIX platform, you can continue to use it (AFM for PostScript and TFM for PCL).

     Note: AFM support is extended only to single-byte PostScript file generation, with the exception of Japanese encoding.

     The encoding schemes supported for the AFM files are:

     ```
     AdobeStandardEncoding
     ExtJIS12-88-CFEncoding
     FontSpecific
     HRoman
     ISOLatinHebrew
     JIS12-88-CFEncoding
     JIS12e-88-CFEncoding
     ```

   - Can scale well. For example, MS Sans Serif does not scale well to a different size, whereas Tahoma does. The reason is that the MS Sans Serif font is a raster font that does not scale well to any size and usually has rounding issues. On the other hand, Tahoma font is a TrueType font that is very similar in visual appearance to the MS Sans Serif font. Additionally, Tahoma is a vector font that can be scaled to any size and rotated to any angle.

12.6.2 Deploying Your Report in Pre-11*g* Version That Uses Motif Tool Kit Mechanism

For fonts with AFM files not readily available on UNIX, or if you encounter any font issues in the report output such as text misalignment, you can convert and generate an AFM file from the Windows TTF file using freely available third party utilities, such as `ttf2pt1`. Do not attempt to convert to a TFM file, as this may not produce reliable results.

To deploy your report on a UNIX platform:

1. Locate the TTF files corresponding to the fonts used in your report. Convert these TTF files to AFM to ensure that you will have the AFM files for the fonts used in your report.

   Use a True Type to Type 1 font converter utility to convert the TTF files to AFM files. For example, `ttf2pt1`.

2. Post-conversion, remove the `.afm` extension in the AFM file name. For example:

Table 12–8 Post Conversion Font File Names

Before Converting	After Converting	After Renaming
ARIALUNI.TTF	arialuni.afm	ArialUnicodeMS

3. Copy the Windows TTF file used in your report to the fonts directory on your UNIX machine. For example, `$ORACLE_HOME/fonts`.
4. Add the path to the TTF file in the `REPORTS_PATH` environment variable. For example, `ARIALUNI.TTF`.
5. Copy the AFM file to the `$ORACLE_HOME/guicommon/tk/admin/AFM` directory. For example, ArialUnicodeMS.
6. Edit the `screenprinter.ppd` file with any text editor.

Note: If you have defined a default printer by including an entry in *$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/uiprint.txt*, then you will have to make the appropriate entries in the printer's PPD file for a PostScript printer or in the HPD file for a PCL printer.

A default printer surface that mimics the screen (`screenprinter.ppd`) is used for formatting if you have not set up a default printer. You will also need to make the necessary font and resolution entries in the `screenprinter.ppd` file, if you have not set up a default printer.

The PPD and HPD files are located at:

- `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin`
- `$ORACLE_HOME/guicommon/tk/admin`

Oracle Reports searches for HPD or PPD files initially in the Oracle Instance location and then in the Oracle Home location.

Refer to Section 10.8.1, "ScreenPrinter" for more information on the `screenprinter.ppd` file.

Ensure that the PPD/HPD file used contains an entry for each AFM or TFM file that you use in your report. PPD/HPD files are configuration files containing printer driver settings and the list of all the fonts supported by the printer.

Navigate to the to the `Font Information` section in the PPD file and add the necessary entries for the font files in the following format:

```
*FONTNAME:ENCODING:VERSION:LOCATION
```

For example:

```
*Font Arial: Standard "(Version 2.76)" Standard ROM
*Font CourierNew: Standard "(Version 2.76)" Standard ROM
```

Ensure that the AFM file name exactly matches the font name specified in the PPD file as Oracle Reports searches for this file based on the font name in the PPD file.

7. Ensure that the `uiprint.txt` has the entry for the appropriate PPD file.

   ```
   printer name:PostScript:2:test:ppd file
   ```

   In this example:

   ```
   printer1:PostScript:2:test:hpljet42.ppd
   ```

8. Edit the `hpljet42.ppd` file with any text editor.

   Note: Copy the PPD file from `ORACLE_HOME/guicommon/tk/admin/PPD` to the following location:

   `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/PPD`

9. Ensure that the PPD/HPD file used contains an entry for each AFM or TFM file that you use in your report.

   Navigate to the `Font Information` section and add the necessary entries for the new AFM files in the following order:

   ```
   *FONTNAME:ENCODING:VERSION:LOCATION
   ```

   For example:

   ```
   *Font ArialUnicodeMS: Standard "(Version 2.76)" Standard ROM
   ```

10. In the `[PDF:Subset]` section, add the following entries to subset the fonts used in your report.

    For example:

    ```
    [ PDF:Subset ]
    "Arial Unicode MS" = "ARIALUNI.TTF"
    ```

    Use the PDF subsetting feature to generate multibyte PDF output from your reports and to ensure your PDF report is portable. Thus, there is not dependency on the machine deploying the report to have the fonts installed.

11. Run the report to PDF and view it.

    ```
    http://mywebserver.com:reports/rwservlet?server=myserver+report=
    "/home/myreports/test.rdf"+authid=hr/hr@mydb+desformat=PDF+destype=cache
    ```

    The PDF should contain the unicode font used in your report. For example, Arial Unicode MS.

12.6.2.1 Troubleshooting Information

If you encounter deployment issues, review the following troubleshooting information:

- If you do not get the correct fonts in the PDF output, set the environment variable `DEBUG_SLFIND` to a log file name (for example, `debug.txt`) and run the report. The font files that are looked up while parsing the PPD file as well as the fonts used will be written to the log file `debug.txt`. Specifically, check for the following:

  1. The PPD file that you modified should be picked up. If it is not picked up it is a configuration issue. Refer to Chapter 9, "Managing Fonts in Oracle Reports".
  2. The AFM files that you have copied to AFM directory should be picked up next.

  See Chapter 11, "Using PDF in Oracle Reports" for more information on `DEBUG_SLFIND`.

12.6.3 Frequently Asked Questions

This section contains frequently asked questions (FAQs) pertaining to deploying a report to Unicode PDF output.

Question

Why is the formatting of my report not correct when using Unicode on UNIX, or why do I see misaligned text in the report?

Answer

If you have used `ttf2pt1` to create the AFM file, `ttf2pt1` has a limitation in that it creates AFM files with metrics information for only the first 256 characters of the font. So, if you are using multiple languages in the report with a Unicode font like Arial Unicode MS, when you create the AFM file for UNIX, there will not be metrics information for the characters beyond the first 256 characters. Oracle Reports uses default metrics information contained in the AFM file when it encounters characters in the report that are not in the AFM file. These default metrics may not match the exact metrics of the characters used in the report. For this reason, formatting will not be correct. To avoid formatting issues when you are using characters beyond the first 256 characters of the font, you can use fixed width fonts where all the characters have the same width.

12.7 Generating PostScript Output

Table 12–9 shows the cross-platform deployment scenario where the destination format is PostScript.

Table 12–9 Cross Platform Deployment - Scenario 3

Development Platform	Deployment Platform	Destination Format
Windows	UNIX	PostScript

This section discusses designing and deploying a report for PostScript output in the following subsections:

- Designing Your Report
- Deploying Your Report
- Frequently Asked Questions

12.7.1 Designing Your Report

To prepare your report before you deploy it on a UNIX platform:

1. Create a new report.
2. Use only those fonts in your report that:
   - Are available on UNIX. All font files that are available on Windows (TTF files) may not be available on UNIX (AFM or TFM files). If you have the correct AFM or TFM font file available on the UNIX platform, you can continue to use it (AFM for PostScript and TFM for PCL).

     Note: AFM support is extended only to single-byte PostScript file generation, with the exception of Japanese encoding.

     The encoding schemes supported for the AFM files are:

     ```
     AdobeStandardEncoding
     ExtJIS12-88-CFEncoding
     FontSpecific
     HRoman
     ISOLatinHebrew
     JIS12-88-CFEncoding
     JIS12e-88-CFEncoding
     ```

   - Can scale well. For example, MS Sans Serif does not scale well to a different size, whereas Tahoma does. This is because MS Sans Serif is a raster font that does not scale well to any size and usually has rounding issues. On the other hand, Tahoma is a TrueType font that is very similar in visual appearance to MS Sans Serif. Additionally, Tahoma is a vector font that can be scaled to any size and rotated to any angle.
   - Do not include Unicode characters. Oracle Reports does not support Unicode character sets in Post Script output on the UNIX platform. As an alternative, you can use either of the following:
     - Oracle Reports PDF output (`desformat=pdf`), which supports multibyte character sets, as discussed in Section 10.6.1, "Multibyte Character Set Printing".
     - Oracle Reports utilities IX and PASTA for font embedding in PostScript output when Oracle Reports is installed and used with Oracle Applications, as discussed in Section 10.6.2, "Overview of IX and PASTA".
3. To have the PostScript output look same on the design platform (Windows) and deployment platform (Unix), the paper size should be same on both the platforms. On Windows, if you want to change the default paper size from Letter to any other size (for example, A4), perform the following steps:
   a. Choose **Settings** > **Control Panel** > **Printers**.
   b. Right-click the default printer and select **Properties**.
   c. Click **Printing Preferences** in the **General** tab.
   d. Click **Advanced** in the **Paper/Quality** tab.
   e. Select the Paper Size and click **OK**.

f. Click **OK** until the main dialog box displays to set the default paper size.

12.7.2 Deploying Your Report

For fonts with AFM files not readily available on UNIX, or if you encounter any font issues in the report output such as text misalignment, you can convert and generate an AFM file from the Windows TTF file using freely available third party utilities, such as `ttf2pt1`. Do not attempt to convert to a TFM file, as this may not produce reliable results.

To deploy your report on a UNIX platform:

1. Locate the TTF files corresponding to the fonts used in your report. Convert these TTF files to AFM to ensure that you will have the AFM files for the fonts used in your report.

   Use a True Type to Type 1 font converter utility to convert the TTF files to AFM files. For example, `ttf2pt1`.

2. Post-conversion, remove the `.afm` extension in the AFM file name. For example:

Table 12–10 Post Conversion Font File Names

Before Converting	After Converting	After Renaming
`arial.ttf`	`arial.afm`	`Arial`

3. Copy the AFM file to the `$ORACLE_HOME/guicommon/tk/admin/AFM` directory.
4. Ensure that the `TK_PRINTER` environment variable or the `PRINTER` environment variable is set to the default printer name. For example, `printer1`.
5. Ensure that `uiprint.txt` has the entry for the appropriate PPD file in the format, `printer name: PostScript:2:test:ppd file`. In this example:

   ```
   printer1:PostScript:2:test:hpljet42.ppd
   ```

6. Edit the file `hpljet42.ppd` using any text editor. Specifically, edit the `DefaultPageSize`, `DefaultPageRegion`, and `DefaultPaperDimension` to change the default paper from Letter to A4, in the following way:

   ```
   …..
   *DefaultPageSize: A4
   …..
   *DefaultPageRegion: A4
   …..
   *DefaultPaperDimension: A4
   ….
   ```

   Note: Copy the PPD file from `ORACLE_HOME/guicommon/tk/admin/PPD` to the following location:

   ```
   $DOMAIN_
   HOME/config/fmwconfig/components/ReportsToolsComponent/<repo
   rts_tools_name>/guicommon/tk/admin/PPD
   ```

7. Ensure that the PPD file used contains an entry for each AFM file that you use in your report.

Navigate to the `Font Information` section and add the necessary entries for the new AFM files in the following order:

```
*FONTNAME:ENCODING:VERSION:LOCATION
```

For example:

```
*Font Arial: Standard "(Version 2.76)" Standard ROM
```

8. Run the report to printer and verify that it is printed to A4 Paper.

```
http://mywebserver.com:reports/rwservlet?server=myserver+report="c:\test.rdf"+a
uthid=hr/hr@mydb+desformat=postscript+destype=cache
```

12.7.3 Frequently Asked Questions

This section contains frequently asked questions (FAQs) pertaining to deploying a report to PostScript output.

Question

Does Oracle Reports support Unicode PostScript file generation?

Answer

Currently, Oracle Reports supports Unicode character sets in PostScript output only on the Windows platform. On UNIX platforms, you can use either of the following:

- Oracle Reports PDF output (`desformat=pdf`), which supports multibyte character sets, as discussed in Section 10.6.1, "Multibyte Character Set Printing".
- Oracle Reports utilities IX and PASTA for font embedding in PostScript output when Oracle Reports is installed and used with Oracle Applications, as discussed in Section 10.6.2, "Overview of IX and PASTA".

Question

Does Oracle Reports embed the font in the PostScript output file?

Answer

Oracle Reports does not embed the font in the PostScript output file. It writes the font name and the metrics that were calculated using AFM files. Therefore, for the report to appear without any font alignment issues, ensure that the necessary fonts are installed on the printer.

Question

The page count of my report varies when run on different installations of UNIX. How can I ensure that the page count of my report is the same regardless of the installation?

Answer

In UNIX, Oracle Reports uses the PPD/HPD file of the default printer in the installation for formatting. The resolution and list of fonts will be picked up from this PPD/HPD files. If there is no default printer setup in the installation, then `screenprinter.ppd` will be used. This PPD file emulates the screen. Earlier versions of Oracle Reports used the `DISPLAY` environment variable instead. Ensure that the two installations *use the same AFM/TFM files and font files*, so that the number of pages of PDF output will be the same.

13

Configuring Destinations for Oracle 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 20, "Creating Advanced Distributions". This chapter explores how Oracle 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 Oracle Reports Services.

It includes the following sections:

- What's New in this Release
- Overview of Output Processing
- Registering Destination Types with the Server
- Submitting Reports to Pluggable Destinations from Oracle Forms Services

13.1 What's New in this Release

12*c* Release (12.2.1.3) provides the Pluggable Destinations from Oracle Forms Services enhancement.

13.1.1 Pluggable Destinations from Oracle Forms Services.

In prior releases, a pluggable destination defined for Reports Server cannot be used when running a report from Oracle Forms Services (specifically from `RUN_REPORT_OBJECT`) because the `REPORT_DESTYPE` property in Oracle Forms Services allows for only a pre-specified set of values.

In Oracle Reports 12*c* Release (12.2.1.3), this limitation is removed: report requests can be submitted to all destinations, including any Oracle Reports-registered pluggable destinations from Oracle Forms Services. This enhancement introduces a new Oracle Forms Services command line parameter: `PLUGDESTYPE`. Oracle Forms Services passes the pluggable destination `DESTYPE` in the `PLUGDESTYPE` parameter as part of the `REPORT_OTHER` property. When this parameter is present, it overrides whatever is specified in `DESTYPE` command line argument (or `REPORT_DESTYPE` for Oracle Forms Services).

See Section 13.4, "Submitting Reports to Pluggable Destinations from Oracle Forms Services"

13.2 Overview of Output Processing

Report output is controlled by the DESTYPE value that you specify at runtime, which, in turn, is determined by the destination output types you have registered in your server configuration file (*rwserver*.conf) using the destination element. See Section A.5.31, "DESTYPE" and Section 7.2.1.5, "destination".

You need not register the following default destinations:

- Cache
- E-mail
- Printer
- File
- FTP
- WebDAV

You *may* need to register the following default destination:

- Oracle Portal: The entry for this destination is created by default in the server configuration file, but it is commented out. To start using this destination, you must uncomment the destination entry, and also provide appropriate property values (for example, the value for the portalUserid property).

You must register any new destination types you create through the Oracle Reports Services Destinations API.

Note: For more information about the destination API, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html).

Configuring destinations is discussed in detail in Chapter 13, "Configuring Destinations for Oracle Reports Services".

You can also define custom output types, such as fax, Oracle's Internet File System (*i*FS), or any new destination type you define using the Oracle Reports Services Destinations API. This API enables you to define new destination types and build handlers to usher your reports to custom destinations.

Note: For more information about the available APIs for Oracle Reports, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html).

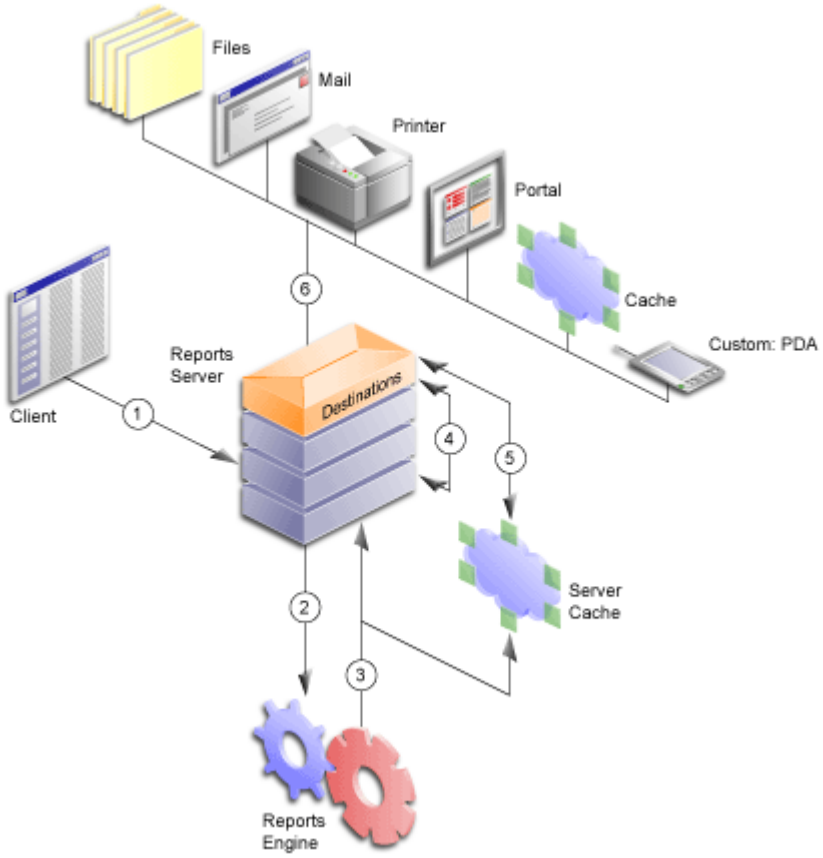
The Oracle Reports Services architecture standardizes the way output is generated and delivered. It takes responsibility for delivering report output to the appropriate destination (through the Reports Server), yet generates output independent of its destination (through the Oracle Reports engine). This 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.

Oracle Reports Services eliminates this tight coupling and its attendant restrictions. The runtime engine now treats all destinations alike. It doesn't 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 13–1 illustrates the main components of the Oracle Reports Services output processing architecture.

Figure 13–1 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.

13.3 Registering Destination Types with the Server

Before Oracle 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 configuration file, *rwserver*.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.

13.3.1 Setting Up a Destination Section in the Server Configuration File

To set up a destination section in the *rwserver*.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):

   ```
   ${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_
   server_name>/rwserver.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.

13.3.2 Entering Valid Values for a Destination

This section outlines the destinations supported by Oracle Reports.

13.3.2.1 Destination destypes and classes

The `destype` and `class` attributes are required for valid registration of a nondefault output type. They specify the destination types and their associated Java classes. The predefined (default) destination types and classes that come with Oracle Reports Services are listed in Table 13–1:

Table 13–1 Standard Destination Types and Classes

Destination	destype	Class
Oracle 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`

> **See Also:** Section A.5.31, "DESTYPE" for examples of 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:** For more information about the available APIs for Oracle Reports, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (`http://www.oracle.com/technetwork/middleware/reports/overview/index.html`).

13.3.2.2 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"
encrypted="no"/>
```

This example defines the values to be associated with a portal user ID. It includes the `encrypted` attribute, 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.xsd` file (*ORACLE_HOME*`\reports\dtd\rwserverconf.xsd`). This is discussed in detail in Chapter 7, "Configuring Oracle 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 is 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.

13.3.3 Example Destination

The following example illustrates a `destination` element for pushing content into Oracle Portal:

```
<destination destype="oraclePortal" class="oracle.reports.server.DesOraclePortal">
  <property name="portalUserid" value="<username_pwd_tnsname_for_logon_to_portal>"
     encrypted="yes"/>
</destination>
```

13.4 Submitting Reports to Pluggable Destinations from Oracle Forms Services

To submit requests to any Oracle Reports-registered pluggable destinations from Oracle Forms Services:

1. Create or install a pluggable destination for Reports Server (refer to the Oracle Reports Plugin Exchange on the Oracle Technology Network (OTN) at http://www.oracle.com/technetwork/middleware/reports/index-085705.html) .
2. In Oracle Forms, create a form and set it up to run a report to Reports Server (refer to *Integrating Oracle Forms 10g and Oracle Reports 10g* white paper at http://www.oracle.com/technetwork/developer-tools/forms/documentation/techlisting10g-085500.html).

   The parameters to submit the request should be set as follows:

   ```
   set_report_object_property(report_id, REPORT_DESTYPE, PLUGDEST);
   ```

```
...
set_report_object_property(report_id, REPORT_OTHER, 'PLUGDESTYPE=pluggable
destination' );
...
```

3. Observe in Reports Server that the report output is sent to the pluggable destination.

When the PLUGDESTYPE parameter is specified, DESTYPE (in Oracle Reports) and REPORT_DESTYPE (in Oracle Forms) will be ignored and the value of PLUGDESTYPE will be used.

14

Configuring and Using the Pluggable Data sources

This chapter describes configuring and using several types of pluggable datasources:

Configuring and Using the JDBC PDS

Configuring and Using Text PDS

Configuring and Using XML PDS

14.1 Configuring and Using the JDBC PDS

The JDBC pluggable data source (PDS) enables you to access any JDBC data 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
- Running a JDBC Report Using Oracle Reports Services
- TroubleShooting Information
- Adding Your Own JDBC Driver

14.1.1 JDBC Configuration File

The `jdbcpds.conf` file is the Oracle Reports JDBC PDS configuration file. It is located in the following directories:

- For Reports Server:
  ${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/
- For Oracle Reports Builder:
  *${DOMAIN_*

HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>

- For Reports Application (in-process Reports Server) deployed in Oracle WebLogic Server: *DOMAIN_HOME*/config/fmwconfig/servers/WLS_REPORTS/applications/reports_*version*/configuration

This file is preconfigured for the:

- Pre-installed drivers; that is, Oracle JDBC Thin, Oracle JDBC OCI (thick), and JDBC-ODBC.
- DataDirect Merant drivers available on Oracle Technology Network, (http://www.oracle.com/technetwork/index.html).

> **Note:** A valid license might be required for the custom JDBC Merant drivers to work here.

You must add or modify relevant entries in the jdbcpds.conf file to include any other JDBC drivers that you want to use.

Oracle 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.

Oracle Reports Builder reads and caches the entries in the jdbcpds.conf when it is invoked. Restart Oracle Reports Builder to view the result of any changes made to the jdbcpds.conf file, for example, 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, and so on.

> **Note:** You can modify or add your driver information in this section.

Example

The following sample illustrates the contents of the jdbcpds.conf file:

```
  <!-- Add or modify the following section for your driver information -->
    <!-- Following drivers are available out-of-box in OracleAS -->

    <jdbcpds>
xmlns="http://xmlns.oracle.com/reports/pdsjdbc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/reports/pdsjdbc
file:c:\orawin/reports/dtd/jdbcpds.xsd"
   >
     <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 14–1 outlines the various attributes that can be associated with a driver.

Table 14–1 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`

`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`
`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 API Reference* for more information on how to extend your JDBC Connection class	`oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling`

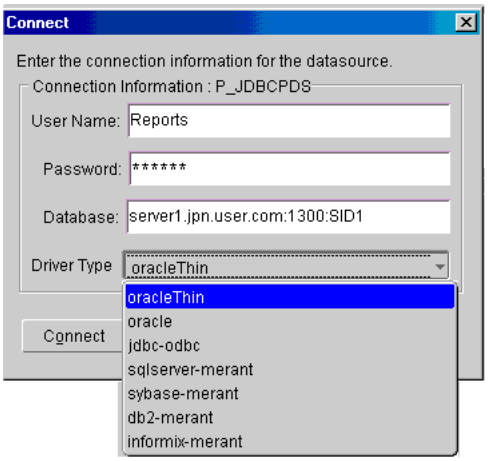
Table 14–1 (Cont.) Driver Attributes

Attribute Name	Description	Sample
`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 14–3, Table 14–4, Table 14–5, Table 14–6, and Table 14–7 for more information on sample connection information.

Figure 14–1 shows a list of all drivers configured in the `jdbcpds.conf` file.

Figure 14–1 JDBC Connect Dialog Box in Oracle Reports Builder



14.1.1.1 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 preconfigured in the `jdbcpds.conf` file. Before you can use SQL Server or Excel with JDBC-ODBC, you must create an ODBC data source. Refer to Windows help, for more information on how to create an ODBC data source.

  Note: Oracle Fusion Middleware provides Merant DataDirect drivers which can also be used to access SQL Server.

14.1.1.2 Installing and Configuring Merant DataDirect Drivers

Oracle provides a set of Merant DataDirect drivers (Version 3.2) that can be downloaded from OTN (`http://www.oracle.com/technetwork/index.html`). The driver configuration file; that is, `jdbcpds.conf` contains relevant entries for the Merant DataDirect drivers. Additionally, the JDBC Connect dialog (Table 14–1) lists the entries for the set of Merant DataDirect drivers provided by Oracle.

However, you must install the appropriate JAR files and specify them in Oracle Reports specific classpath entries, in order to make them available to Oracle Reports Builder and Oracle Reports Services

The drivers provided by Oracle for use with Oracle Fusion Middleware / Oracle Developer Suite are:

- Sybase Driver
- DB 2 Driver
- SQL Server Driver
- Informix Driver
- Custom Driver

You can also install and configure a Custom Driver for use with Oracle Fusion Middleware 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 Fusion Middleware and Oracle Developer Suite directory.
2. Include an entry in `REPORTS_CLASSPATH` to make the files available to Oracle Reports Builder and Oracle Reports Services.

   Note: The `REPORTS_CLASSPATH` variable is located in the `reports.sh` file for all UNIX platforms.

   Refer to the relevant driver in this section for information on the required JAR files.

   a. **Oracle 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 UNIX 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 `${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/` directory. Refer to Table 14–1 for more information on the parameters. Refer to the relevant driver in this section for an example.

14.1.1.2.1 Sybase Driver

1. Install the relevant JAR files in your Oracle Fusion Middleware 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 Oracle Reports Builder and Oracle Reports Services.

   Note: The `REPORTS_CLASSPATH` variable is located in the `reports.sh` file for all UNIX platforms.

   a. **Oracle 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 UNIX 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_INSTANCE\config\ReportsServerComponent\server_name` directory. Refer to Table 14–1 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>
   ```

14.1.1.2.2 DB 2 Driver

1. Install the relevant JAR files in your Oracle Fusion Middleware and Oracle Developer Suite directory.

   JAR files required: `YMutil.jar`, `YMdb2.jar`, and `YMbase.jar`

2. Include an entry in `REPORTS_CLASSPATH` to make the files available to Oracle Reports Builder and Oracle Reports Services.

   Note: The `REPORTS_CLASSPATH` variable is located in the `reports.sh` file for all UNIX platforms.

   a. **Oracle 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 UNIX 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_INSTANCE\config\ReportsServerComponent\server_name` directory. Refer to Table 14–1 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>
   ```

14.1.1.2.3 SQL Server Driver

1. Install the relevant `.jar` files in your Oracle Fusion Middleware 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 Oracle Reports Builder and Oracle Reports Services.

   Note: The `REPORTS_CLASSPATH` variable is located in the `reports.sh` file for all UNIX platforms.

   a. **Oracle 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 UNIX 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 `${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/` directory. Refer to Table 14–1 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>
   ```

14.1.1.2.4 Informix Driver

1. Install the relevant JAR files in your Oracle Fusion Middleware 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 Oracle Reports Builder and Oracle Reports Services.

   Note: The `REPORTS_CLASSPATH` variable is located in the `reports.sh` file for all UNIX platforms.

   a. **Oracle 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 UNIX 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 `${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/` directory. Refer to Table 14–1 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>
   ```

14.1.1.2.5 Custom Driver

Any driver that is not provided by Oracle must be installed and configured. For example, you can use BEA JDBC drivers if you have license. To install and configure a custom driver, complete the following steps:

1. Install the relevant JAR files in your Oracle Fusion Middleware and Oracle Developer Suite directory.
2. Include an entry in `REPORTS_CLASSPATH` to make the files available to Oracle Reports Builder and Oracle Reports Services.

   Note: The `REPORTS_CLASSPATH` variable is located in the `reports.sh` file for all UNIX platforms.

   Jar files required: Refer to the relevant driver documentation.

   a. **Oracle 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 UNIX 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 `${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/` directory. Add relevant driver configuration information to the `jdbcpds.conf` file. Refer to Table 14–1 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

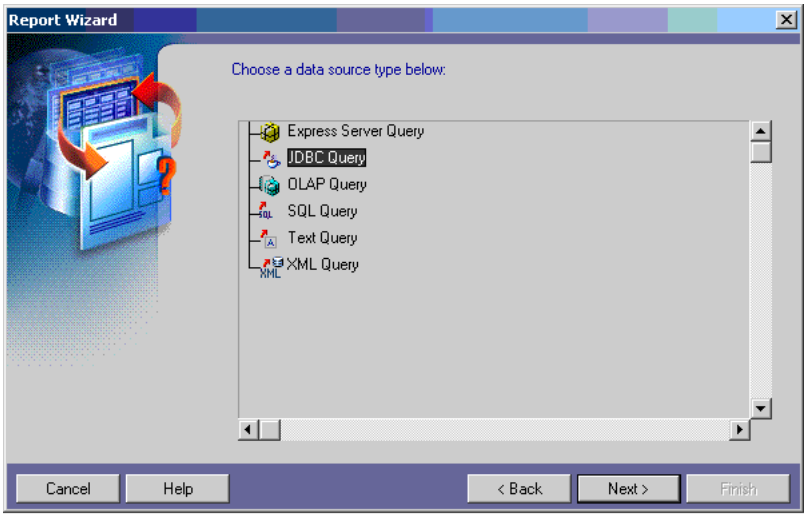
14.1.2 Defining and Running a JDBC Query

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 Oracle 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 *Oracle Reports online Help*.

Figure 14–2 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.

Figure 14–3 Calling a stored procedure

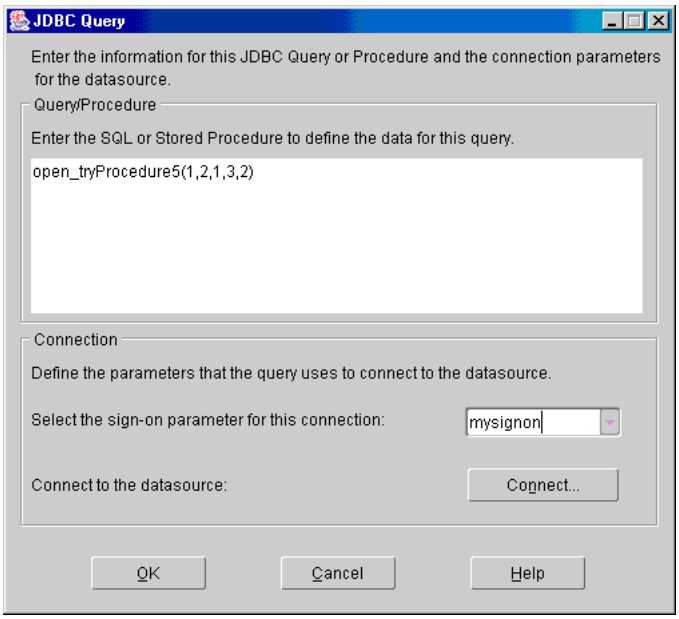


Table 14–2 Specifying an Excel Data Source

Query (Single Worksheet)	Query (Multiple Worksheets)
`SELECT * FROM [SHEET1$]` or `SELECT COL1, COL2, ...COLn FROM [SHEET1$]`	`SELECT * FROM [WORKSHEETNAME$]`
Where `SHEET1$` is the name of a .xls file	Where `[WORKSHEETNAME$]` is the name of the worksheet
Where the first worksheet row value is taken as a column name for the query	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.	**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 name is `P_JDBCPDS` (see Section A.7.10, "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 Oracle Reports Services.

   b. Enter the connection information (user name, password, and database name) for the driver type. Refer to Table 14–3, Table 14–4, Table 14–5, Table 14–6, and Table 14–7 for sample connection information.

   c. Select the driver type. The driver list is displayed based on the values entered in the `jdbcpds.conf` file.

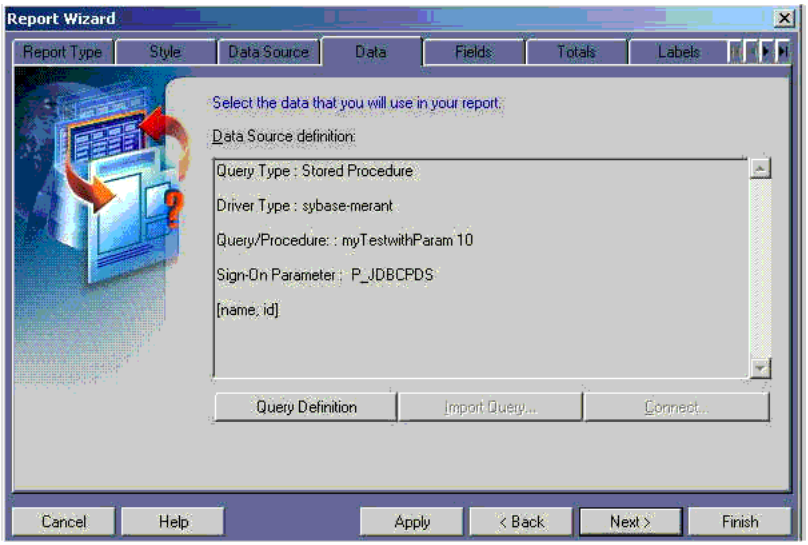
   d. Click **Connect** to gain access to the database using the new sign-on. The connect string formed internally is a combination of:

      * The `connectString` driver attribute (Table 14–1) defined in the `jdbcpds.conf` file

* The connection information supplied in the Connect dialog will be used to fill the database name portion of the `connect string`.

7. Click **OK** to execute the JDBC query.
8. The Reports Wizard displays the query description (Figure 14–4).

Figure 14–4 Query Description



9. Follow the steps in the wizard to define the layout and to run the report based on your JDBC query.

14.1.2.1 Sample Connection Information

Table 14–3, Table 14–4, Table 14–5, Table 14–6, Table 14–7,Table 14–8, and Table 14–9 lists sample connection information for use with:

- Pre-installed drivers; that is, Oracle JDBC Thin, Oracle JDBC OCI (thick), and JDBC-ODBC.
- DataDirect Merant drivers available on Oracle Technology Network, (`http://www.oracle.com/technetwork/index.html`).

Table 14–3 Oracle Thin Driver

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. Example: `server1.example.com:1300:session1`

Table 14–4 Oracle Thick Driver

Property	Value
Username	`Reports`
Password	`Welcome`
Database	`n123` where `n123` is a tnsname entry in the `tnsnames.ora` file

Table 14–5 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 14–6 Sybase

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 number of the TCP/IP port. Example: `server1.example.com:1300`

Table 14–7 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`

Table 14–8 SQL Server

Property	Value
Username	`Reports`
Password	`Welcome`

Table 14–8 (Cont.) SQL Server

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. Example1: `server1:1654`

Table 14–9 Informix

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. `InformixServer name`: The Informix server name. `Database name`: The database name you are connected to. Example2: `server_name:2003;InformixServer=myinformix_server;DatabaseName=scott/tiger@mydb`

14.1.3 Running a JDBC Report Using Oracle Reports Services

When you run a report containing 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 Reports Builder during 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:

- `userid` is the value for connecting the SQL query to the Oracle database. You need not specify the `userid` if your report does not have a SQL query or a `REF CURSOR` query.
- `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.

  The default sign-on parameter name P_JDBCPDS will be used if you have not specified a name in the JDBC Query dialog box while designing the report in Reports Builder.

14.1.4 TroubleShooting Information

This section lists:

- JDBC PDS error messages (Error Messages)

- JDBC query troubleshooting (Trace Information).

14.1.4.1 Error Messages

Table 14–10, Table 14–11, and Table 14–12 lists troubleshooting information related to the JDBC PDS.

Table 14–10 Error Messages Related to the Database Connection

Error Message	Cause	Action
`Connection class {0} can't be loaded`	Invalid connection class specified 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 14–11 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.

Table 14–11 (Cont.) Error messages Related to Executing the Data Source

Error Message	Cause	Action
`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.
`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 14–12 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 `${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/` directory.	Ensure that the `jdbcpds.conf` file is available in the `file is available in the ORACLE_INSTANCE\config\ReportsServerComponent\server_name` 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 specified 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.

14.1.4.2 Trace Information

Use the detailed trace information 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 14–1 for sample design-time trace output.

See Example 14–2 for sample run-time trace output.

For more information on the location of the `logging.xml` file, see Section 24.2.5, "Tracing Report Execution"

Sample trace output

Example 14–1 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.example.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>oracle
</driverType><connectionClass>oracle.reports.plugin.datasource.jdbcpds.JDBCConnect
ionHandling</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 14–2 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</driverType>
<connectionClass>oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling</
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
```

14.1.5 Adding Your Own JDBC Driver

Note: Oracle Reports exposes the PDS API and also contains a tutorial that describes in detail how to implement or customize your own PDS. For more information, refer to Oracle Reports Java API Reference, available on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html). 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 JDBC PDS are:

- Configuring the jdbcpds.conf File
- Installing the Driver's JAR Files

14.1.5.1 Configuring the jdbcpds.conf File

For information on how to configure the `jdbcpds.conf` file, refer to Section 14.1.1, "JDBC Configuration File".

14.1.5.2 Installing the Driver's JAR Files

For information on how to install the driver's JAR files, refer to Section 14.1.1.2.5, "Custom Driver".

14.2 Configuring and Using Text PDS

Text PDS is configured and available out-of-the-box for Oracle Reports.

14.2.1 Text Configuration File

The `textpds.conf` file is the Oracle Reports Text PDS configuration file which can be changed, added, or deleted. The file is located at the following directories:

- For Reports Server:

  ${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/textpds.conf

- For Oracle Reports Builder:

${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/textpds.conf

- For Reports Application (in-process Reports Server) deployed in Oracle WebLogic Server:
  DOMAIN_HOME/config/fmwconfig/servers/WLS_REPORTS/applications/reports_*version*/configuration

Example

The following example illustrates the contents of `textpds.conf` file:

```
<!--XML section.Please edit this section to give your file format information-->
<textPDS
    xmlns="http://xmlns.oracle.com/reports/pdstext"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.oracle.com/reports/pdstext
file:c:\orawin/reports/dtd/textpds.xsd"
    >

<!--Data definition for Apache Log file format-->
  <fileFormat  name = "Apache Log File" comment = "#" delimiter = "default"
               type ="variable" nullValue = "-">
    <columnInfo>
<column    name = "Remote Host"      type = "string" />
      <column    name = "Remote Log Name"  type = "string"/>
      <column    name = "Remote User"      type = "string"/>
      <column    name = "time"             type = "date"    cellWrapper = "\[]"
                 pattern = "dd/MMM/yyyy:hh:mm:ss zzz"/>
      <column    name = "Request"          type = "string"  cellWrapper =
"&amp;quot;"/>
      <column    name = "status"           type = "number"/>
      <column    name = "bytes"            type = "number"/>
    </columnInfo>
  </fileFormat>

<!--Data definition for Comma Delimited file format-->
 <fileFormat  name = "Example Variable Width Comma Delimited" comment = "#"
              delimiter = "," type = "variable">
     <columnInfo>
      <column    name = "var_cd_col1"     type = "string" />
      <column    name = "var_cd_col2"     type = "string"/>
      <column    name = "var_cd_col3"     type = "string"/>
     </columnInfo>
  </fileFormat>

  <!--Data definition for Fixed Width format-->
  <fileFormat  name = "Example Fixed Width Space Delimited" comment = "#"
               delimiter = ""  type = "fixed">
     <columnInfo>
      <column    name = "fw_sd_col1"     type = "string" startpos = "1"  width =
"13"/>
      <column    name = "fw_sd_col2"     type = "number" startpos = "14" width =
"10"/>
      <column    name = "fw_sd_col3"     type = "date"   startpos = "25" width =
"10"/>
    </columnInfo>
 </fileFormat>
</textPDS>
```

Table 14–13 outlines the various values that can be associated with a File Formats.

Table 14–13 File Format Attributes

Attribute Name	Description
`name`	A File Format name, this name appears in the Data Definition drop down list
`Comment`	A Comment character, this will be used in the DataSource.
`Delimiter`	A Column Delimiter. The Data Source file contains data in rows. Each row has fields or tokens corresponding to the Columns specified in the configuration file. Each field or token will be separated by a Column delimiter (See cellWrapper)
`type`	A File Format Type which can be either a fixed or variable. In case of `variable` file format, the column width is variable. In case of `fixed` file format, the column width is fixed and is specified using the attributes `startpos` and `width` for each column.
`nullValue`	This is a null value character.

Table 14–14 outlines the various values that can be associated with a Column.

Table 14–14 Column Attributes

Attribute Name	Description
`name`	This is the name of the Column Heading
`type`	This is the Column Data type. There are three data types that are supported: ■ `String` ■ `Number` ■ `Date`
`cell Wrapper`	This is an optional attribute. If specified, this character will override the File Format delimiter for the column for which it is specified.
`pattern`	This is an optional attribute. Currently it is used only for date fields. It specifies the pattern in which date field is to be expected.

14.3 Configuring and Using XML PDS

XML PDS is configured and available out-of-the-box.

14.3.1 XML PDS Configuration File

The `xmlpds.conf` file is the Oracle Reports XML PDS configuration file. It is located at the following directories:

- For Reports Server:

  `${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/`

- For Oracle Reports Builder:

${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/

- For Reports Application (in-process Reports Server) deployed in Oracle WebLogic Server: *DOMAIN_HOME*/config/fmwconfig/servers/WLS_REPORTS/applications/reports_*version*/configuration

> **Note:** The only parameter that can be changes is the columnLength. The default size of the column is 4000 bytes

Example

The following example illustrates the contents of xmlpds.conf file:

```
<xmlpds
   xmlns="http://xmlns.oracle.com/reports/pdsxml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.oracle.com/reports/pdsxml
file:c:\orawin/reports/dtd/xmlpds.xsd"
   >
     <columnInfo>
         <columnLength>4000</columnLength>
     </columnInfo>
</xmlpds>
```

14.4 Specifying the encoding of an XML PDS Report

When you create a report against an XML data source, you must ensure that the encoding of the data source and its DTD matches the encoding of Reports Builder.

For example, when you create an XML report against a table encoded in a Japanese character set, the group element name is encoded in Japanese. To match the data source, you should encode the group's element name in the DTD in Japanese. The XML and DTD files can be in any encoding that supports Japanese, such as Shift_JIS, EUC-JP, or UTF-8.

If you do not match the XML data source and DTD encoding to the Reports Builder encoding, you will see the following error:

```
ERR-063001 xxx.dtd null
```

> **Note:** You will not see this error if you use a XML schema instead of a DTD.

To avoid this problem, ensure that both the XML data source and DTD for your XML report use the same encoding that you have in the character encoding part of the NLS_LANG environment variable in effect for your Reports Runtime.

For example, if NLS_LANG=JAPANESE_JAPAN.JA16SJIS for your Reports Runtime, then both your XML data source and DTD should use Shift_JIS.

15

Securing Oracle Reports Services

The celebrated openness of the Internet brings with it concerns about controlling who has access to what confidential company information. Oracle Reports 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 contains the following sections:

15.1 Introduction to Oracle Reports Security

This section introduces security features and concepts in Oracle Reports.

This section discusses the following topics:

15.1.1 Overview

In addition to working with Oracle Reports 12*c* Release (12.2.1.3), you can now run in a Single Sign-on environment using Oracle Access Manager 11*g* (OAM) as the

authentication server. For more information see, Chapter 17, "Configuring and Administering Oracle Single Sign-On".

Oracle Reports 12*c* Release (12.2.1.3) uses a standards-based Java EE security model through Oracle Platform Security Services. This provides a flexible, simple to administer security mechanism. It can be used with standalone Oracle Reports install or any Forms-Reports combination. The policy store and the identity store used for authentication and authorization can be standard JAZN-XML based or any LDAP server, including Oracle Internet Directory through JAZN-LDAP, providing flexibility.

> **Note:** JAZN-XML is an XML file which is configured by the user to use as an id store and/or policy store.

Oracle Reports 12*c* Release (12.2.1.3) accomplishes authentication through Single Sign-On, Oracle Internet Directory, Embedded ID Store, and JAZN-XML File-based ID Store. For authorization, Oracle Reports 12*c* Release (12.2.1.3) supports Oracle Internet Directory, File-based, and Portal-based methods. In prior releases, Reports Server authentication was restricted to use only Oracle Internet Directory. If you want to revert to the security mechanism of prior releases, you can do so as described in Section 6.3.1.1, "Switching to Oracle Portal Security". If you want to use Single Sign-On without implementing data source security or Oracle Portal, refer to Chapter 17, "Configuring and Administering Oracle Single Sign-On".

Alternatively, you might have your own application for launching reports with its own login mechanism and user/group repository, or have your own mechanism for protecting data sources (for example, you might choose to use a different LDAP server to store user and group information). In this case, Oracle Reports Services provides interfaces that allow you to integrate it with these non-Oracle components, as described in Section 15.14, "Security Interfaces".

15.1.2 Resources Protected

Oracle Reports Services encompasses functionality for three main areas of security:

- Application Security (that is, controlling access to the report application, where users launch report requests)
- Resource Security (that is, controlling access to reports and Reports Servers)
- Data Source Security (that is, for controlling access to a particular database)

15.1.2.1 Application Security

Generally, users must log on to an application or site (for example, your own corporate Web site, Oracle WebCenter) 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 log in and, optionally, data source security. In a typical configuration, the user logs on through OracleAS Single Sign-On to gain access to a report application, where they access and run their reports.

15.1.2.2 Resource Security

Resource security ensures that only authorized users or groups execute a specific report. It also keeps users or groups from accessing particular Reports Servers for the execution of the report. Certain servers might be reserved for a particular group of users, or 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 through the specified Reports Server to the specified destination, then the user's privileges to the data source accessed by the report must be ascertained.

Optionally, or for backward compatibility, you can configure Oracle Portal to provide resource security for reports and Reports Servers out of the box. In a typical configuration, the administrator or developer specifies which users and groups could access which reports and Reports Servers from Oracle Portal.

15.1.2.3 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 Oracle Reports Services:

- You can associate data source connection information with a Single Sign-On user. When using OAM 11*g* server, the user must create a resource in the OID using batch loading. See Section 17.3.3.2.2, "Batch Loading". For more information about batch loading, see *Oracle Fusion Middleware Administrator's Guide for 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 in to 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` for OracleAS Single Sign-On (OSSO) server and the necessary connection parameters (for example, `USERID,SSOCONN`). This functionality is much the same as it was in previous releases of Oracle Reports Services. For a complete discussion of URL syntax, refer to Section 18.1, "The Reports URL Syntax". For a complete discussion of key mapping, refer to Section 18.14, "Using a Key Map File".

> **See Also:** Section 17.3.3, "Enabling and Disabling Data Source Security"
>
> Section 17.3.3.1, "SSOCONN"

15.1.3 Credential Store

A Credential Store is the repository of security data that certify the authority of entities used by Java 2, JavaEE and ADF applications. Applications can use the credential

store, a single, consolidated service provider to store and manage the credentials securely.

A domain includes one credential store. Application-specific credentials are supported and migrated to credentials in the domain credential store when the application is deployed. Thus, all servers and all applications deployed in a domain use a common credential store, the domain credential store.

Oracle Reports 12*c* Release (12.2.1.3) uses credential store to store a password as a key. You can also use the credential store to configure database connection information for `jobStatusRepository` and `jobRepository` elements.

For example:

Portal password is stored in the reports credential map with key in the following syntax:

```
"portalpasswd_DomainName_InstanceName"
```

> **Note:** You must create credentials under the Reports folder as the server accesses credentials from this folder in CSF.

15.1.3.1 Credential Types

Oracle Platform Security supports the following types of credentials according to the data they contain:

- A *password* Credential encapsulates a user name and a password
- A *generic* credential encapsulates any customized data or arbitrary token, such as a symmetric key.

In Credential Store Framework (CSF), a credential is uniquely identified by a map name and a key name. Typically, the map name corresponds with the name of an application and all credentials with the same map name define a logical group of credentials, such as the credentials used by the application. All map names in a credential store must be distinct. If the credential store is intended to be the repository of X.509 certificates, it is recommended the use of an Oracle Wallet or a Java keystore. The credential store does not allow the storage of end-user digital certificates.

> **Note:** CSF keys are stored in `rwserver.conf` and `rwservlet.properties` file.

For more information on how to manage credentials in a domain credential store through Oracle Enterprise Manager, see Section 6.3.8, "Managing Credentials".

For more information about Wallet-Based and LDAP-Based Credential Stores and Configuring the Credential Store, see Securing Applications with Oracle Platform Security Services.

15.2 Out-of-the-Box Behavior

Out-of-the-Box reports is secured using OPSS security. User can configure to OID based security using WLST commands.

15.3 Authentication in Oracle Reports

This section describes authentication features, tasks, and concepts that are specific to Oracle Reports.

It discusses the following topics:

- Single Sign-On Authentication
- Non-SSO Authentication
- Authentication Scenarios for JPS-Based Security
- Authentication Scenario for Portal-Based Security

Authentication Methods

Oracle Reports 12*c* Release (12.2.1.3) supports the following authentication methods:

- Singe Sign-On. See Single Sign-On Authentication.
- Non-SSO, including the following:
  - Oracle Internet Directory (`rwsec`, or JPS-OID configured)
  - Embedded ID store (in-process servers)
  - JAZN-XML File-Based ID store (standalone servers)

> **Note:** For more information about non-SSO authentication methods, see Non-SSO Authentication.

The following table summarizes the authentication methods for JPS-based security that Oracle Reports supports.

Table 15–1 Authentication Methods for JPS-Based Security

Type of Reports Server	Oracle Internet Directory	WebLogic ID Store	Single Sign-On	File-Based
In-process servers	Yes	Yes	Yes	No
Standalone servers	Yes	No	Yes	Yes

The following table summarizes the authentication methods for Portal-based security that Oracle Reports supports.

Table 15–2 Authentication Methods for Portal-Based Security

Type of Reports Server	Authentication Based on Oracle Internet Directory	Single Sign-On
In-process servers	Yes	Yes
Standalone servers	Yes	Yes

15.3.1 Single Sign-On Authentication

Oracle Reports Services applications can now run in a single sign-on environment using Oracle Access Manager 11*g* (OAM) and Oracle Internet Directory (OID) to eliminate the need for additional or different logins to access many applications during the same user session. Oracle Reports Services applications in Oracle FMW 12*c* Release (12.2.1.3)can be protected by the following authentication servers in the single sign-on environment:

- Oracle Access Manager (OAM) 11*g*

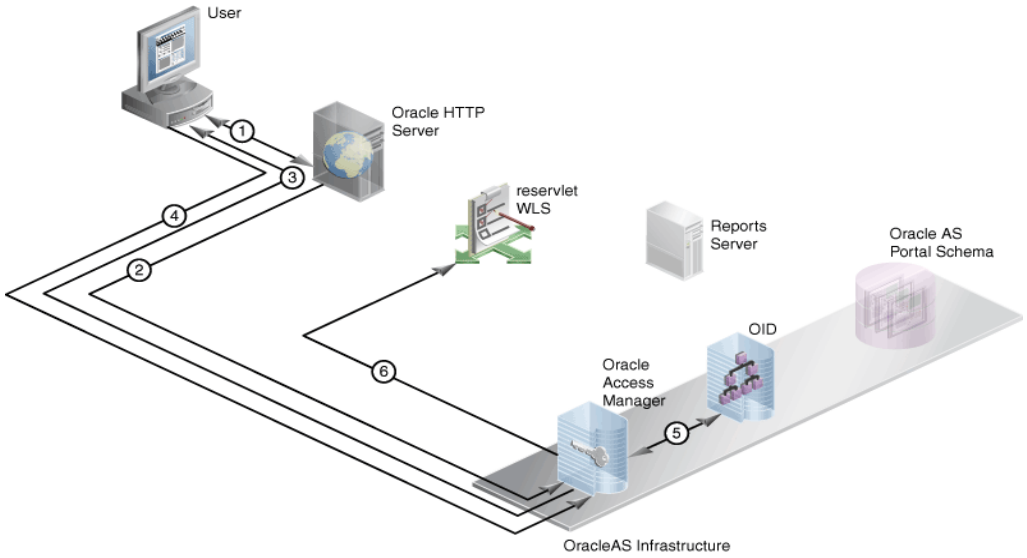
During the installation of authentication servers, users can choose to authenticate their Reports Applications using one of these authentication servers. It is required that these authentication servers are configured to use Oracle Internet directory as the backend Identity Store. Authentication servers are designed to work in Web environments where multiple Web-based applications are accessible from a browser. Without an authentication server, each user must maintain a separate identity and password for each application they access. Maintaining multiple accounts and passwords for each user is unsecure and expensive.

Oracle Access Manager 11*g* is a Java Platform, Enterprise Edition (Java EE)-based enterprise-level security application that provides restricted access to confidential information and centralized authentication and authorization services. Oracle Access Manager 11*g*, a component of Oracle Fusion Middleware 11*g*, is a Single Sign-On solution for authentication and authorization.

15.3.1.1 Authentication Flow with Oracle Access Manager (OAM) 11g

Figure 15–1 describes the authentication process with Oracle Access Manager as the authentication server.

Figure 15–1 Authentication Process with OAM 11g Server



1. User requests the report (through a URL).
2. Oracle HTTP Server routes the request to the OAM server.
3. OAM server sends a page requesting the user name and password.
4. The credentials, provided by the user, are sent to the OAM server.

5. Oracle Access Manager (OAM) verifies the credentials with Oracle Internet Directory.
6. If the user is authenticated, OAM server passes the "user authenticated" message to **`rwservlet`**.

   If you used `SSOCONN` in your URL, `rwservlet` checks the Single Sign-On key against Oracle Internet Directory to see if it already has been mapped to a data source connection string (for example, `scott/tiger@my_or_db`).

   If you used `SSOCONN`, and 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.

   Note: When your Reports application is authenticated with OAM 11*g* server, even if the Single Sign-On parameter is set to `No`, the OAM authentication page is displayed and not the Reports authentication page.

   If you used `SSOCONN` but Oracle Internet Directory does not already contain a connection string for the key, then Oracle Reports raises a "key does not exist" error message.

15.3.2 Non-SSO Authentication

If any of the non-SSO authentication methods is used (based on Oracle Internet Directory, File-based in case of JPS-based security, and Embedded ID store), then any user accessing a secured instance of the Reports Server is challenged to identify themselves by `rwservlet` or Reports clients through their own authentication mechanism.

Table 15–3 Non-SSO Authentication Methods

ID Store	Authentication
Oracle Internet Directory (`rwsec`, or JPS-OID configured)	Authentication against Oracle Internet Directory
Embedded ID store (in-process servers)	Authentication against embedded ID store of WebLogic Server
JAZN-XML File-based ID store (standalone servers)	Authentication against file-based ID store

Because the HTTP 1.0 protocol is stateless (that is, each call to the server is effectively independent of all others), users may want 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 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 $DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/cgicmd.dat. %S forces users to enter their username and password each time the report is called. See Section 18.14, "Using a Key Map File".

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 as an attribute of the element cookie in the rwservlet.properties file.

For example, you can specify the cookie element in the rwservlet.properties file as follows:

```
<cookie cookieexpire="30" encryptionkey="reports"/>
```

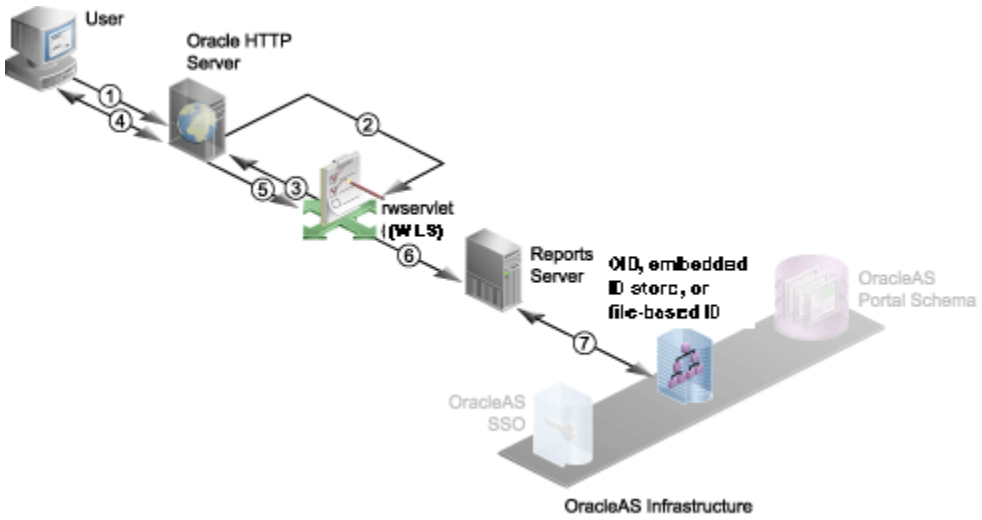
When rwservlet receives a job request, it compares the time saved in the cookie with the current system time. If the time is longer than the number of minutes defined in the environment variable (for example, 30 minutes), the cookie is rejected and the user is challenged to provide authentication information.

See Also: Section 7.3, "Oracle Reports Servlet Configuration File" for more information about the COOKIEEXPIRE parameter and the rwservlet.properties file.

15.3.2.1 Report Request Flow with Non-SSO (Oracle Internet Directory-Based, File-Based, or Embedded ID Store)

In this scenario, the report request is sent to a secured Reports Server with Single Sign-On disabled. Non-SSO authentication methods include Oracle Internet Directory-based, File-based, and Embedded ID store. In this case, rwservlet or a JSP report might be called through the use of a bookmark or from an Oracle Portal component.

Figure 15–2 Authentication Process Without Single Sign-On



The following numbered steps map to the numbers in Figure 15–3:

1. **User requests the report (through a URL)**.

   The user must somehow gain access to the URL that launches the report request (for example, through a link on a Web page or a bookmark), and choose the URL.

2. **Oracle HTTP Server routes the request to `rwservlet` deployed on Oracle WebLogic Server**.

3. **`rwservlet` asks for user credentials (that is, user name and password)**.

   `rwservlet` checks for the `AUTHID` parameter in 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 Oracle Reports `AUTHID` cookie. (If the report is launched from Oracle Portal, `AUTHID` is added to the URL automatically.) If neither the `AUTHID` parameter nor an Oracle Reports `AUTHID` cookie is found, `rwservlet` sends the System Authentication page to the Oracle HTTP Server, to display to the user.

4. **Oracle HTTP Server displays the login page to the user, and the user provides user name and password**.

   On the login page, the user must supply a user name and password. This information is stored in an Oracle Reports `AUTHID` cookie for future reference.

5. **User name and password are passed on to `rwservlet`**.

   If only partial data source credentials are provided in the URL (for example, `USERID=scott@orqa`), the Database Authentication page displays with the partial credentials shown. 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 through the `DBAUTH` parameter in the `rwservlet.properties` file. If no data source credentials are provided, the Database Authentication page does not display and it is assumed the report does not require a data source.

   See Also: Section 7.3, "Oracle Reports Servlet Configuration File" for more information about the `DBAUTH` parameter and the `rwservlet.properties` file.

The data source credentials are stored in an Oracle Reports `USERID` cookie for future reference. Note that pluggable data source (PDS) credentials are not stored in Oracle Reports `USERID` cookies.

6. **`rwservlet` forwards user name and password to Reports Server**.

   `rwservlet` constructs a command line with the necessary information from the previous steps and passes it to Reports Server.

7. **Reports Server authenticates the user (that is, verifies the user name and password) against the ID Store**.

   Reports Server validates the user credentials against the ID store (Oracle Internet Directory, embedded ID store or file-based Oracle Internet Directory). If the validation check fails for any reason, then an error condition is returned to the user and the process terminates.

15.3.3 Authentication Scenarios for JPS-Based Security

This section discusses the following authentication scenarios:

- If Reports is using JPS security, JPS-OID for security policies, and an embedded ID store
- If Reports is using JPS security and JPS-OID as ID store

> **Note:** By default, an in-process server uses the embedded ID store of Oracle WebLogic Server as the ID store and the `system-jazn-data.xml` file as the policy store. Standalone servers use the `system-jazn-data.xml` file as both ID store and DB as the policy store.

15.3.3.1 If Reports is using JPS security, JPS-OID for security policies, and an embedded ID store

It is recommended that you move users in your current ID store, such as embedded ID store, to Oracle Internet Directory, which is an LDAP-based ID store. Subsequently, you can map users to application roles. For information about moving users to Oracle Internet Directory, see the section "Migrating Identities Manually" in the *Securing Applications with Oracle Platform Security Services*. For information about mapping users to application roles, see Mapping Users to Application Roles.

15.3.3.2 If Reports is using JPS security and JPS-OID as ID store

You must map users in Oracle Internet Directory to the default application roles. For information about mapping users to application roles, see Mapping Users to Application Roles.

> **Note:** In the above authentication scenarios, if Single Sign-On is enabled, the Single Sign-On screen is displayed. If Single Sign-On is disabled, the Reports sysauth screen is displayed. In either case, users are authenticated against Oracle Internet Directory. If you have not moved your users to Oracle Internet Directory, then users are authenticated against the embedded ID store for in-process servers. For standalone servers, such users are authenticated against the file-based ID store.

15.3.4 Authentication Scenario for Portal-Based Security

If you are using Portal-based security, Oracle Internet Directory-based authentication is used.

You can map users to application roles. For information about mapping users to application roles, see Mapping Users to Application Roles.

> **Note:** In the above authentication scenarios, if Single Sign-On is enabled, the Single Sign-On screen is displayed. If Single Sign-On is disabled, the Reports sysauth screen is displayed. In either case, users are authenticated against Oracle Internet Directory.

15.4 Authorization in Oracle Reports

If you are using JPS-based security, you can use either Oracle Internet Directory or Database method for authorization. If you are using Portal-based security, the Portal-based authorization is used.

In the case of JPS-based security, an in-process server uses `database.xml` as the policy store, by default. Hence, Reports policies are stored in `database.xml` under the reports application entry. Users are authorized based on this policy store. For standalone servers, all the policies are stored in the database.`xml` file and authorization is done against these policies.

> **Note:** The authorization process involves checking whether a particular user is in the ID store used by JPS. If Single Sign-On is used for authentication, Ensure that the same users are configured in the ID store used by JPS. Alternatively, ensure that JPS points to the ID store used by Single Sign-On. Otherwise, authorization does not work.

The following table summarizes the supported authorization methods if Oracle Reports uses JPS-based security.

Table 15–4 Authorization Methods for JPS-Based Security

Types of Report Server	Oracle Internet Directory	File Based
In-process	Yes	Yes
Standalone	Yes	Yes
Database	Yes	Yes

If Portal-based security is configured, the following authorization methods are used.

Table 15–5 Authorization Method for Portal-Based Security

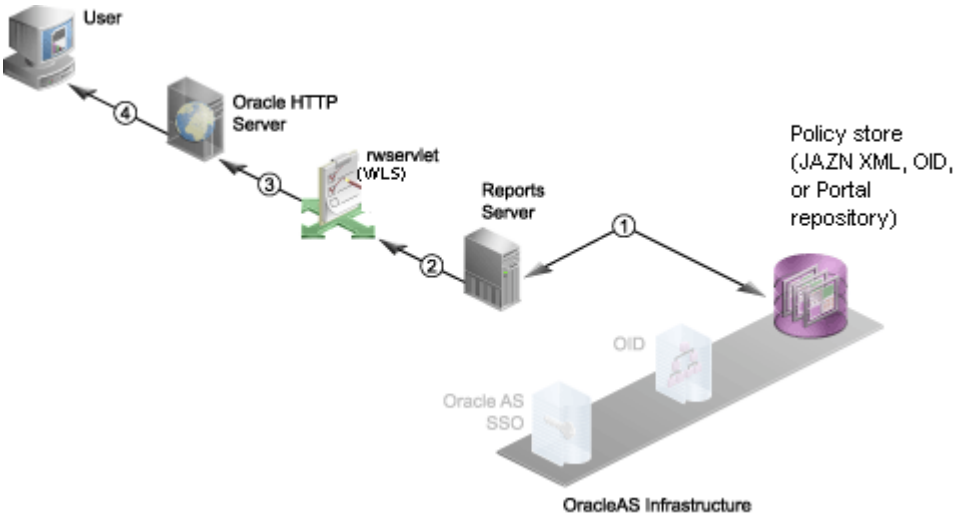
Types of Report Server	Authorization
In-process	Portal-based
Standalone	Portal-based

Note: If Oracle Portal is configured to perform authorization, and the report request is launched from within Oracle Portal rather than `rwservlet`, Oracle Reports will similarly validate the user's privileges on the report before running it. Even for unauthenticated (`PUBLIC`) users viewing public pages, Oracle Reports Services verifies that the `PUBLIC` user account has appropriate privileges on the report.

15.4.1 Authorization Process

Authorization occurs after a user is authenticated using Single Sign-On or Non-SSO (Oracle Internet Directory-based, File-based in case of JPS-based security, Database and Embedded ID store) methods. Once the user is authenticated, the report request must go through the authorization process, as shown in Figure 15–3.

Figure 15–3 Authorization Process Flow



The following numbered steps map to the numbers in Figure 15–4:

1. **Reports Server validates the user privileges against the policies defined in the Policy Store.**

   Reports Server validates the user privileges against the policies defined in Policy Store (Database, LDAP, or Portal repository) by the user.

   Reports Server checks whether the user has the necessary privileges to run the report on the parameters specified in the Policy Store. If the validation check fails for any reason, then an error condition is returned to the user and the process terminates.

   Note: If the user is executing `rwservlet` Web commands such as `showjobs` and `getserverinfo`, instead of executing a report, Reports Server verifies and authorizes the user based on Policy Store settings.

2. **If the user is authorized to execute the report, Reports Server executes the report request and passes the report output to `rwservlet`.**

Reports Server delegates the job to an engine that accesses the data source, retrieves the data, and formats the report.

3. **Report output is passed to Oracle HTTP Server**.
4. **Report output is passed to the user**.

   The completed output is sent to the specified destination. Depending upon the destination, the output may be served back to the browser (as shown in Figure 15–4), sent to a printer, stored in a file for future reference, sent to an FTP server, and so on.

15.4.2 Additional Step When Using JPS for Authorization

Reports policies are granted to application roles. You must associate all users in your ID store (embedded ID store of Oracle WebLogic Server or an external Oracle Internet Directory) with one of the Reports application roles.

You must add the `oracle.security.jps.enterprise.user.class` property in the `jps-config-jse.xml` file.

In Enterprise Manager, you can complete this task as follows:

1. Navigate to the **WebLogic Domain** menu.
2. Choose **Security** > **Application Roles**.

   The **Application Roles** page is displayed. In this page, you can map users to application roles.

Alternatively, you can complete this task by manually editing the `$DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml` file. This step is required if you want to use JPS to authorize your users in Oracle Internet Directory.

Search for the "reports" application in the XML file and add a user in the members section. For example, to add a user called `orcladmin`, add:

```
<member>
<class>weblogic.security.principal.WLSUserImpl</class>
<name>orcladmin</name>
</member>
```

15.4.3 Defining Security Policies for Reports

Out-of-the-box, default users, roles, and permissions are already created. As administrator, you can specify the reports to which a particular user has access by defining a security policy for each report. In the security policy, you can also specify the server, destination name (`desname`), destination type (`destype`), and other parameters. The security policy is checked when the user provides the user name and password.

15.4.3.1 Defining Security Policies for JPS-Based Security

Refer to Section 6.3.2, "Defining Security Policies for Reports" to use Oracle Enterprise Manager to update the report security policies.

15.4.3.2 Defining Security Policies for Portal-Based Security

For Portal-based security, you can create a security policy in Oracle Portal. See the "Securing Oracle Portal" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

15.4.4 Defining Security Policies for Directories for JPS-Based Security

In certain cases, you will want to give a particular user access to multiple related reports. Rather than specify a security policy for each report, you can collect all the reports in a single directory, then specify a security policy for the directory. Again, the security policy is checked when the user provides the user name and password.

Refer to Section 6.3.3, "Defining Security Policies for Directories" to use Oracle Enterprise Manager to update the directory security policies.

15.4.5 Defining Security Policies for Web Commands for JPS-Based Security

You can also specify the Oracle Reports Servlet (`rwservlet`) Web commands to which a particular user/role has access by creating security policies for each Web command. The security policy is checked when the user provides the user name and password.

Refer to Section 6.3.4, "Defining Security Policies for Web Commands" to use Oracle Enterprise Manager to update the Web command security policies.

15.4.6 Defining Read/Write Access to Directories

As administrator, you can specify read/write access for Reports Server, Reports Application (in-process Reports Server), or Oracle Reports Runtime to directories. This feature only checks whether the Reports Server, Reports Application (in-process Reports Server), or Oracle Reports Runtime is authorized to read from or write to a specified directory, and is unrelated to the security policies for users/roles, which check the user name and password.

15.4.7 Searching Application Policies in Enterprise Manager

Application policies are the authorization policies that an application uses for controlling access to its resources. You can enter search keyword for principals or permissions to query application security grants. You can use an application stripe to search if the application uses a stripe that is different from the name of the application.

To search for application policies in Enterprise Manager, complete the following steps:

1. Log in to Enterprise Manager.
2. From the **Weblogic Domain** menu, select **Security** > **Application Policies**.

   The **Application Policies** page is displayed.
3. Check the **Select Application Stripe to Search** option.
4. In the drop-down menu, select `reports`.
5. In the **Principal** field, enter the name of the principal.
6. In the **Permissions** field, enter the permissions.
7. Click the right arrow button to search application security grants.

15.4.8 Searching Application Roles in Enterprise Manager

Application roles are the roles used by security-aware applications that are specific to the application. These roles are seeded by applications in WebLogic Domain policy store when the applications are registered.

To search for application roles in Enterprise Manager, complete the following steps:

1. Log in to Enterprise Manager.

2. From the **Weblogic Domain** menu, select **Security** > **Application Roles**.

   The **Application Roles** page is displayed.

3. Check the **Select Application Stripe to Search** option.
4. In the drop-down menu, select `reports`.
5. In the **Role Name** field, enter the name of the application role to search.
6. Click the right arrow button to search application roles.

15.5 End-to-End Security Scenarios

This section describes end-to-end security scenarios that involve both authentication and authorization.

The following table describes JPS-based security scenarios.

Table 15–6 JPS-Based Security Scenarios

Security Scenario	Description
JPS-OID Authorization with Single-Sign-On Authentication for Reports Servlet	
This scenario involves the following: ▪ Single Sign-On for authentication ▪ JPS-OID for authorization (policies)	To use this combination of authentication and authorization, complete the following steps: 1. Enable Single Sign-On. See Enabling and Disabling Single Sign-On. 2. Enable JPS-based security by editing reports server config file. 3. Ensure that all users that are present in the Oracle Internet Directory used by Single Sign-On are in the ID store used by JPS. Alternatively, configure JPS to point to the ID store used by Single Sign-On. 4. Add the following property in the `jps-config-jse.xml` file: `<property name="oracle.security.jps.enterprise.user.class" value="weblogic.security.principal.WLSUserImpl"/>` 5. Configure JPS Oracle Internet Directory as a policy store. Alternatively you can use the database as policy store which is the default policy store. See Configuring an External Oracle Internet Directory as Policy Store When Using JPS-Based Security. 6. Create security policies. Refer to Section 6.3.2, "Defining Security Policies for Reports" to use Oracle Enterprise Manager to update the report security policies defined in Oracle Internet Directory. 7. Map users to application roles. For more information about mapping users to application roles, see Mapping Users to Application Roles.
JPS-OID Authorization with JPS-OID as ID Store for Other Reports Clients	

Table 15–6 (Cont.) JPS-Based Security Scenarios

Security Scenario	Description
This scenario involves the following: ▪ JPS-OID for authentication ▪ JPS-OID for authorization (policies)	To use this combination of authentication and authorization, complete the following steps: 1. Enable JPS-based security by editing reports server config file. 2. Add the following property in the `jps-config-jse.xml` file: `<property name="oracle.security.jps.enterprise.user.class" value="weblogic.security.principal.WLSUserImpl"/>` 3. Configure JPS-OID as an ID store. See Configuring External Oracle Internet Directory as ID Store When Using JPS-Based Security. 4. Configure JPS-OID as a policy store. Alternatively you can use the database as policy store which is the default policy store. See Configuring an External Oracle Internet Directory as Policy Store When Using JPS-Based Security. 5. Create security policies. Refer to Section 6.3.2, "Defining Security Policies for Reports" to use Oracle Enterprise Manager to update the report security policies defined in Oracle Internet Directory. 6. Map users to application roles. For more information about mapping users to application roles, see Mapping Users to Application Roles.
JAZN-XML Authorization with Single Sign-On Authentication for Reports Servlet	
This scenario involves the following: ▪ Single Sign-On for authentication ▪ JAZN-XML for authorization (policies)	To use this combination of authentication and authorization, complete the following steps: 1. Enable Single Sign-On. See Enabling and Disabling Single Sign-On. 2. Enable JPS-based security. by editing reports server config file. 3. Ensure that all users that are present in the Oracle Internet Directory used by Single Sign-On are in the ID store used by JPS. Alternatively, configure JPS to point to the ID store used by Single Sign-On. 4. Add the following property in the `jps-config-jse.xml` file: `<property name="oracle.security.jps.enterprise.user.class" value="weblogic.security.principal.WLSUserImpl"/>` 5. Create security policies. Refer to Section 6.3.2, "Defining Security Policies for Reports". 6. Map users to application roles. For more information about mapping users to application roles, see Mapping Users to Application Roles. 7. If the `system-jazn-data.xml` file is used as the policy store, search for the "reports" application in the `system-jazn-data.xml` file. Database is used as default policy store. It is not recommended to change this to file based policy store (`system-jazn-data.xml`). To use JPS to authorize users in Oracle Internet Directory, add the corresponding users in the member section of the `system-jazn-data.xml` file. See Section 15.4.2, "Additional Step When Using JPS for Authorization".
JAZN-XML Authorization with JPS-OID Authentication for Other Reports Clients	

Table 15–6 (Cont.) JPS-Based Security Scenarios

Security Scenario	Description
This scenario involves the following: ▪ JPS-OID for authentication ▪ JAZN-XML for authorization (policies)	To use this combination of authentication and authorization, complete the following steps: 1. Enable JPS-based security by editing reports server config file. 2. Add the following property in the `jps-config-jse.xml` file: `<property name="oracle.security.jps.enterprise.user.class" value="weblogic.security.principal.WLSUserImpl"/>` 3. Configure JPS-OID as an ID store. See Configuring External Oracle Internet Directory as ID Store When Using JPS-Based Security. 4. Create security policies. Refer to Section 6.3.2, "Defining Security Policies for Reports" to update the report security policies defined in Oracle Internet Directory. 5. Map users to application roles. For more information about mapping users to application roles, see Mapping Users to Application Roles. 6. If the `system-jazn-data.xml` file is used as the policy store, search for the "reports" application in the `system-jazn-data.xml` file. Database is used as default policy store. It is not recommended to change this to file based policy store (`system-jazn-data.xml`). To use JPS to authorize users in Oracle Internet Directory, add the corresponding users in the member section of the `system-jazn-data.xml`. See Section 15.4.2, "Additional Step When Using JPS for Authorization".

The following table describes Portal-based security scenarios.

Table 15–7 Portal-Based Security Scenarios

Security Scenario	Description
Portal-Based Authorization with Single-Sign-On Authentication for Reports Servlet	
This scenario involves the following: ▪ Single Sign-On for authentication ▪ Portal-based authorization (policies)	To use this combination of authentication and authorization, complete the following steps: 1. Enable Single Sign-On. See Enabling and Disabling Single Sign-On. 2. Enable Portal based security by editing reports server config file. If you have enabled JPS-based security, switch to Portal-based security. 3. Create security policies in Oracle Portal. For more information about creating security policies in Oracle Portal, see the Securing Oracle Portal chapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*. 4. Map users to application roles. For more information about mapping users to application roles, see Section 16.1, "Creating Reports Users and Named Groups"
Portal-Based Authorization with Oracle Internet Directory as ID Store for Other Reports Clients	

Table 15–7 (Cont.) Portal-Based Security Scenarios

Security Scenario	Description
This scenario involves the following: ■ Oracle Internet Directory for authentication ■ Portal-based for authorization (policies)	To use this combination of authentication and authorization, complete the following steps: 1. Configure Oracle Internet Directory as an ID store. See Configuring External Oracle Internet Directory as ID Store. 2. Enable Portal based security by editing reports server config file. If you have enabled JPS-based security, switch to Portal-based security. 3. Create security policies in Oracle Portal. For more information about creating security policies in Oracle Portal, see the Securing Oracle Portal chapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*. 4. Map users to application roles. For more information about mapping users to application roles, see Section 16.1, "Creating Reports Users and Named Groups"

15.6 Recommended Production Scenario for JPS-Based Security

For JPS-based security, the following production scenario is recommended:

- For authentication, Oracle Internet Directory as ID store, or Single Sign-On
- Oracle Internet Directory or Database as Policy store for authorization

If you are not using Single Sign-On, use Oracle Internet Directory for authentication.

15.7 Recommended Production Scenario for Portal-Based Security

For Portal-based security, the following production scenario is recommended:

- Oracle Internet Directory as ID store for authentication, or Single Sign-On
- Portal-based authorization

If you are not using Single Sign-On, use Oracle Internet Directory for authentication.

15.8 Managing Users and Security Policies

This section describes how to manage users and security policies for in-process servers and standalone servers.

It discusses the following topics:

- Adding Users to WebLogic Embedded ID Store for In-Process Servers
- Adding Policies to Policy Store for In-Process Servers
- Mapping Users to Application Roles
- Adding Users to system-jazn-data.xml for Standalone Servers
- Adding Policies to Policy Store for Standalone Servers

15.8.1 Adding Users to WebLogic Embedded ID Store for In-Process Servers

To add users to the ID store for an in-process server, complete the following steps:

1. Navigate to the WebLogic Server Administration Console.
2. From the **Domain Structure** window on the left pane of the console, select **Security Realms** > **myrealm**.

The **Settings for myrealm** page is displayed.

3. Click the **Users and Groups** tab.
4. Click **New** to add users.

   The **User Properties** page is displayed.
5. Enter the parameters, and select the **DefaultAuthenticator** from the **Provider** drop-down list.
6. Click **OK**.

15.8.2 Adding Policies to Policy Store for In-Process Servers

For more information about security policies, see Section 6.3.2, "Defining Security Policies for Reports".

15.8.3 Mapping Users to Application Roles

After configuring the users, you must map users present in the ID store to one or more application roles. You can configure an application role for an in-process server either through Oracle Enterprise Manager or manually.

In Enterprise Manager, you can complete this task as follows:

1. Navigate to the **WebLogic Domain** menu.
2. Choose **Security** > **Application Roles**.

   The **Application Roles** page is displayed. In this page, you can map users to application roles.

If the `system-jazn-data.xml` file is used as the policy store, you can add the following under the reports entry in the `system-jazn-data.xml` file to configure users to application roles manually:

```
<app-role>
   <name>rw_administrator</name>
   <display-name>Reports Administrator</display-name>
   <class>oracle.security.jps.service.policystore.ApplicationRole</class>
   <members>
      <member>
      <class>weblogic.security.principal.WLSUserImpl</class>
      <name>weblogic</name>
     </member>
   </members>
</app-role>
```

For a sample `system-jazn-data.xml` file, see Sample system-jazn-data.xml File

15.8.4 Adding Users to system-jazn-data.xml for Standalone Servers

If the `system-jazn-data.xml` file is used as the ID store, you can users to ID store for a standalone server by completing the following steps:

1. Add users to the `system-jazn-data.xml` file by replacing `<jazn-realm/>` with the following:

   ```
   <jazn-realm>
      <realm>
        <name>jazn.com</name>
          <users>
   ```

```
        <user>
          <name><weblogic</name>
            <credentials>!passwd</credentials>
         </user>
        </users>
      </realm>
    </jazn-realm>
```

For a sample `system-jazn-data.xml` file, see Sample system-jazn-data.xml File

15.8.5 Adding Policies to Policy Store for Standalone Servers

For more information about security policies, see Section 6.3.2, "Defining Security Policies for Reports".

15.9 Configuring External Oracle Internet Directory and Reassociating Reports

This section describes how to configure external Oracle Internet Directory for in-process servers and standalone servers and to reassociate Reports with Oracle Internet Directory and another Portal.

It discusses the following topics:

- Configuring External Oracle Internet Directory for In-Process Servers
- Reassociating Reports with Oracle Internet Directory
- Reassociating Oracle Reports to Oracle Portal
- Configuring External Oracle Internet Directory for Standalone Servers

15.9.1 Configuring External Oracle Internet Directory for In-Process Servers

You can migrate from the default embedded ID store of WebLogic Server to an external Oracle Internet Directory to configure the ID store and Policy store settings. Note that configuration of an external Oracle Internet Directory is a post-installation procedure.

15.9.1.1 Configuring External Oracle Internet Directory as ID Store When Using JPS-Based Security

If you are using JPS-based security, you can configure an external Oracle Internet Directory as ID store through the Oracle WebLogic Server Administration Console.

To configure an external Oracle Internet Directory as an ID store through Oracle WebLogic Server, complete the following steps:

1. Navigate to the WebLogic Server Administration Console.
2. From the **Domain Structure** window, select **Security Realms**.

   The **Summary of Security Realms** page is displayed.
3. Select a Realm from the Realms table.
4. From the settings for the realm page, click the **Providers** tab.
5. Select **New** from the list of Authentication Providers.
6. Enter a name in the **Name** field. From the **Type** drop-down list, select **OracleInternetDirectoryAuthenticator**, and click **OK**.

7. Select the new authenticator, and set the **Control Flag** to **Sufficient**.
8. Select the **Provider Specific** tab, and enter valid values in the appropriate fields.
9. Select the default Authenticator and set the **Control Flag** to **OPTIONAL**.
10. Click **Save**.
11. Restart the Admin Server.
12. Select the **Security Realm** > **Users and Groups**. Ensure that all users of external Oracle Internet Directory are seen on this page.

Now, users trying to access the in-process servers are authenticated based on the users specified in the external Oracle Internet Directory.

15.9.1.2 Configuring an External Oracle Internet Directory as Policy Store When Using JPS-Based Security

If you are using JPS-based security, you can configure an external Oracle Internet Directory as policy store though Oracle Enterprise Manager.

To configure the policy store in Oracle Enterprise Manager, complete the following steps:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the WebLogic domain.
3. From the **WebLogic Domain** menu, select **Security** > **Security Provider Configuration**.
4. Click **Configure**.
5. Provide the Oracle Internet Directory and JPS root node details and click **OK**.

> **Note:** If the JPS root node does not exist on Oracle Internet Directory, you must create it. See *Oracle Fusion Middleware Administrator's guide for Oracle Internet Directory*.

The policies configured in the `system-jazn-data.xml` file are migrated to the external Oracle Internet Directory.

15.9.2 Reassociating Reports with Oracle Internet Directory

If you have configured an external Oracle Internet Directory, you must reassociate Reports to map to the new Oracle Internet Directory.

To map Reports to associate with the new Oracle Internet Directory using WLST complete the following steps:

```
$MW_HOME/oracle_common/common/bin/wlst.sh
connect("weblogic","welcome1", "localhost:7001")

# help for reports server commands
help('reports_server')
help('associateStandaloneServer')
help('dissociateStandaloneServer')
help('associateInprocessServer')
help('dissociateInprocessServer')

# Associate Standalone Server
```

```
associateStandaloneServer(serverName='reports_server_name', oidHost='oid_host_
name', oidPort='oid_port', oidUserName='oid_user_name', oidPassword='oid_
password', sslEnabled='oid_port_ssl_enabled')

eg
associateStandaloneServer(serverName='repsvr1', oidHost='oidhost.example.com',
oidPort='3060', oidUserName='cn=orcladmin', oidPassword='welcome1',
sslEnabled='false')

# Dissociate Standalone Server
dissociateStandaloneServer(serverName='reports_server_name', oidHost='oid_host_
name', oidPort='oid_port', oidUserName='oid_user_name', oidPassword='oid_
password', sslEnabled='oid_port_ssl_enabled')

eg
dissociateStandaloneServer(serverName='repsvr1', oidHost='oidhost.example.com',
oidPort='3060', oidUserName='cn=orcladmin', oidPassword='welcome1',
sslEnabled='false')

# Associate Inprocess Server
associateInprocessServer(wlsName='managed_server_name', oidHost='oid_host_name',
oidPort='oid_port', oidUserName='oid_user_name', oidPassword='oid_password',
sslEnabled='oid_port_ssl_enabled')

eg
associateInprocessServer(wlsName='WLS_REPORTS', oidHost='oidhost.example.com',
oidPort='3060', oidUserName='cn=orcladmin', oidPassword='welcome1',
sslEnabled='false')

# Dissociate Inprocess Server
dissociateInprocessServer(wlsName='managed_server_name', oidHost='oid_host_name',
oidPort='oid_port', oidUserName='oid_user_name', oidPassword='oid_password',
sslEnabled='oid_port_ssl_enabled')

eg
dissociateInprocessServer(wlsName='WLS_REPORTS', oidHost='oidhost.example.com',
oidPort='3060', oidUserName='cn=orcladmin', oidPassword='welcome1',
sslEnabled='false')
exit()
```

15.9.3 Reassociating Oracle Reports to Oracle Portal

Ensure that you have associated Reports with the Oracle Internet Directory. To reassociate Oracle Reports to new Oracle Portal, complete the following steps:

1. Log in to Oracle Enterprise Manager.
2. Navigate to the WebLogic Domain Home page.
3. From the **WebLogic Domain** menu, select **Security** > **Credentials**.
4. Add Portal Credentials in the Credential Store.

   Add a new key value pair in the `reports` map. For example, add a key as `hrportalPasswdKey` and key value as the portal schema password.

   See Section 6.3.8, "Managing Credentials".
5. Edit reports server configuration and enter valid values for Portal Connection, Portal Username, and Portal Password Key.

Note: Reassociating Oracle Reports to Oracle Portal affects only the particular server on which the changes are made. You must repeat the procedure to reassociate each Reports server in the Domain Home to an Oracle Portal.

15.9.4 Configuring External Oracle Internet Directory for Standalone Servers

You can migrate from the default ID store (JAZN-XML) to an external Oracle Internet Directory to configure the ID store and Policy store settings. Note that the configuration of an external Oracle Internet Directory is a post-installation step.

To configure an external Oracle Internet Directory as an ID store or policy store, you must modify the `$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml` file manually.

15.9.4.1 Configuring External Oracle Internet Directory as ID Store

To configure an external Oracle Internet Directory as an ID store, modify the `$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml` file as described in the following procedure.

Note: This is just an example. You must replace the example values provided in the entries with your install-specific values.

1. Under `<jpsContext name="default">`, add the following:

```
<serviceInstanceRef ref="idstore.oid"/>
```

Comment out the following:

```
<!--serviceInstanceRef ref="idstore.xml"/-->
```

2. Under `<serviceInstances>`, add the following entries:

```
<serviceInstance name="idstore.oid" provider="idstore.ldap.provider">
             <property name="subscriber.name"
 value="dc=us,dc=abc,dc=com"/>
             <property name="idstore.type" value="OID"/>
             <property name="cleartext.ldap.credentials"
 value="cn=password"/>
             <property name="ldap.url"
 value="ldap://abc.us.com:389"/>
             <extendedProperty>
                 <name>user.search.bases</name>
                 <values>
                    <value>cn=users,dc=us,dc=abc,dc=com</value>
                 </values>
             </extendedProperty>
             <extendedProperty>
                 <name>group.search.bases</name>
                 <values>
                   <value>cn=groups,dc=us,dc=abc,dc=com</value>
                 </values>
             </extendedProperty>
             <property name="username.attr" value="uid"/>
             <propperty name="groupname.attr" value="cn"/>
 </serviceInstance>
```

3. Under `< serviceProviders>`, add the following:

```
<serviceProvider type="IDENTITY_STORE" name="idstore.ldap.provider"
 class="oracle.security.jps.internal.idstore.ldap.LdapIdentityStoreProvider">
        </serviceProvider>
             <description>Prototype LDAP-based ID store</description>
```

15.9.4.2 Configuring External Oracle Internet Directory as Policy Store

To configure an external Oracle Internet Directory as a policy store, modify the `$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml` file as described in the following procedure.

> **Note:** This is just an example. You must replace the example values provided in the entries with your install-specific values.

1. Under `<jpsContext name="default">` add the following:

```
<serviceInstanceRef ref="policystore.ldap"/>
```

Comment out the following:

```
 <!--serviceInstanceRef ref="policystore.xml"/-->
```

2. Under `<serviceInstances>`, add the following:

```
<serviceInstance provider="ldap.policystore.provider" name="policystore.ldap">
            <property value="OID" name="policystore.type"/>
            <property name="security.principal" value="cn=orcladmin" />
            <property name="security.credential" value="password" />
            <property value="cn=PRDomain"
name="oracle.security.jps.farm.name"/>
            <property value="cn=sta796_sa_root"
name="oracle.security.jps.ldap.root.name"/>
            <property value="ldap://abc.us.com:389" name="ldap.url"/>
        </serviceInstance>
```

3. Under `<serviceProviders>`, add the following:

```
<serviceProvider type="POLICY_STORE" name="ldap.policystore.provider"
 class="oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider">
             <description>Prototype LDAP-based ID store</description>
        </serviceProvider>
```

4. Save and restart WLS_REPORTS.

15.10 Forms and Reports Security Recommendations

The following security model is recommended for applications using Reports and Forms.

If Reports is Using Portal-Based Security

- It is recommended that Forms and Reports are associated to same Oracle Internet Directory.

  For more information see, Configuring External Oracle Internet Directory for In-Process Servers, and Configuring External Oracle Internet Directory for Standalone Servers

- It is recommended that you enable Single Sign-On
- Enable Single Sign-On by editing reports servlet configuration

If Reports is using JAZN security

If Reports is using JPS-based security, by default, an in-process server uses the embedded ID store of WebLogic Server as the ID store and an Database Policy store. A standalone server uses JAZN-XML based ID store. Forms uses Oracle Internet Directory based authentication for security. In this scenario:

- It is recommended that you configure Reports to use Oracle Internet Directory-based ID store. Forms and Reports should use the same Oracle Internet Directory.

  For more information about configuring external Oracle Internet Directory, see, Configuring External Oracle Internet Directory for In-Process Servers, and Configuring External Oracle Internet Directory for Standalone Servers
- Database is used as default policy store. This is recommended. You can also migrate to using OID based policy store if needed.
- It is recommended that you enable Single Sign-On
- Enable Single Sign-On by editing reports servlet configuration

15.11 Intermediate-level Security for Forms and Reports

Oracle Reports 12*c* Release (12.2.1.3) provides new security measures for reports run from Oracle Forms Services in non-secure mode:

- Oracle Reports allows you to generate random and non-sequential job IDs to make it impossible to predict the job ID for a particular job. See Section 18.9.2, "Generating Random and Non-Sequential Job IDs".

  Prior to 12*c* Release (12.2.1.3), Oracle Reports generated sequential job IDs, making it easy to predict the job ID. This meant that unauthorized or malicious users could potentially view the job output using GETJOBID through `rwservlet` to obtain job output that belongs to another user.
- Web commands (`rwservlet` keywords) are now categorized for added security:
  - End user Web commands: GETJOBID, KILLJOBID, SHOWAUTH, SHOWJOBID
  - Administrator Web commands: DELAUTH, GETSERVERINFO, KILLENGINE, PARSEQUERY, SHOWENV, SHOWJOBS, SHOWMAP, SHOWMYJOBS. AUTHID is required to run administrator commands
  - `L0`: no Web commands allowed.
  - `L1`: only end user Web commands allowed (GETJOBID, KILLJOBID, SHOWAUTH, SHOWJOBID).
  - `L2`: administrator Web commands (DELAUTH, GETSERVERINFO, KILLENGINE, PARSEQUERY, SHOWENV, SHOWJOBS, SHOWMAP, SHOWMYJOBS) are also allowed. AUTHID is required to run administrator commands.
  - `NO` (for backward compatibility with `DIAGNOSTIC=NO` in 10*g* `rwservlet.properties`). No Web commands allowed.
  - `YES` (for backward compatibility with `DIAGNOSTIC=YES` in 10*g* `rwservlet.properties`). Administrator Web commands (DELAUTH,

GETSERVERINFO, KILLENGINE, PARSEQUERY, SHOWENV, SHOWJOBS, SHOWMAP, SHOWMYJOBS) are also allowed. AUTHID is required to run administrator commands.

Note: For L2 Web command access, you do not need to pass the authid. The authid parameter is required only for the STOPSERVER command irrespective of the webcommandaccess value.

Administrators are allowed to run both end user and administrator Web commands. For a non-secure Reports Server, the user ID and password for administrators can be set in the identifier element of the Reports Server configuration file.

The new webcommandaccess parameter in the Oracle Reports Servlet (`rwservlet`) configuration file (`rwservlet.properties`) defines access levels for executing `rwservlet` keywords (Web commands). These values can be set using Oracle Enterprise Manager, as described in Section 6.3.4, "Defining Security Policies for Web Commands".

15.12 Database Proxy Authentication

Oracle Reports 12*c* Release (12.2.1.3) provides support for database authentication using proxy users:

- Additional security through control of users that are allowed to connect to the database through Oracle Reports.
- Scalability, through reuse of a single database connection.

You can use the pre-11*g* security mechanism of Oracle Internet Directory integration for authentication without using Oracle Platform Security Services in either of the following ways:

15.12.1 Using DAS and Editing the Server Configuration File

Step 1. Using Delegated Administration Service (DAS): DAS is used to create a resource in OID only in case of OSSO 10g server if it is available. When OAM 11g is used as the authentication server, see Batch Loading in Section 17.3.3.2, "Populating Oracle Internet Directory" for creating a resource in OID. For more information see, *Oracle Fusion Middleware Administrator's guide for Oracle Internet Directory*.

To define a user in Oracle Internet Directory:

1. Determine the configuration values of Oracle Internet Directory for application identity store. For example:

```
host:port: stbpo44.oracle.com:3060
user name: cn=orcladmin
password: welcome1
```

2. Use the DAS URL if you want to add new users or use the existing users on Oracle Internet Directory. For example

```
http://stbpo44.oracle.com:7788/oiddas
```

Step 2. Configure the standalone Report Server to use the RWSecurity to enable the non-Oracle Platform Security Services way of using Oracle Internet Directory (as in 11.1.2/11.1.1).

1. Navigate to the Reports Server component.
2. Check out `rwserver.conf`.
3. Enable the `RWSecurity` element.
4. Make sure that the security id for the `job` element reflects the security id of the `RWSecurity` element.
5. Save `rwserver.conf`.
6. Add the following line in `rwservlet.properties`:

   ```
   <singlesignon>no</singlesignon>
   ```

7. Restart the Reports Server component (see Chapter 5, "Starting and Stopping Oracle Reports Services").
8. Test authentication using the following URL:

   `http://`*`host`*`:`*`port`*`/reports/rwservlet/showjobs?server=`*`reports_server_name`*

15.12.2 Configuring Proxy User Authentication in the Database

You can configure proxy user authentication in the database as follows:

1. Create a proxy user in the database. All clients connecting to the middle tier share this user connection.
2. Run the following:

   ```
   CREATE USER proxy_user1 IDENTIFIED BY welcome1;
   ```

3. Assign the following minimum privileges to proxy_user1:

   ```
   GRANT CONNECT, RESOURCE, CREATE ANY DIRECTORY, DROP ANY DIRECTORY TO
   proxy_user1
   ```

   The actual user already exists in the database.

4. Log in as the actual user, and assign privileges to the proxy user to connect to the database on behalf of the actual user.

   a. `ALTER USER scott GRANT CONNECT THROUGH proxy_user1;`

   b. Or you can define the roles that the proxy user can connect to the database as.

   c. `ALTER USER scott GRANT CONNECT THROUGH proxy_user WITH ROLE admin;`

   d. Repeat Step A or Step C for all actual database users.

The proxy user with minimum privileges is created. However, this proxy user can connect as the actual user with the assigned role. The middle tier can connect to the database as the proxy user first, and then connect as an actual user through the proxy user account.

15.12.3 Obtaining Proxy Access Information

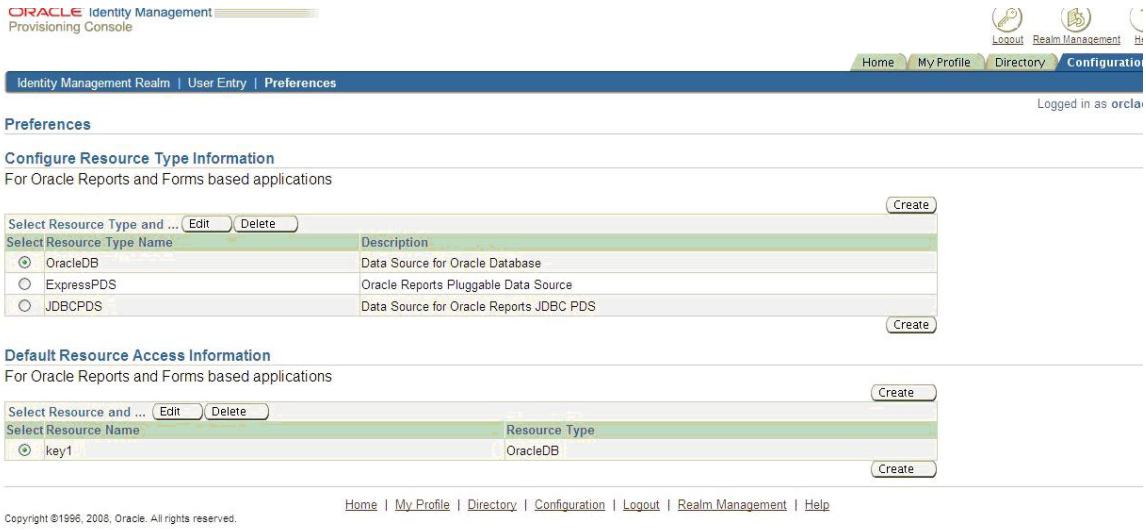
You can configure a new resource in Oracle Internet Directory in the user RAD or default RAD. You can use this key to obtain proxy access information, including user name, password , and database information from Oracle Internet Directory. This information allows you to connect to the database.

To create a key in the default RAD, complete the following steps:

1. Log in to the Oracle Internet Directory configuration page (`http://oidhost:port/oiddas`).

2. Click the `Configuration` tab.
3. Click `Preferences`.
4. In the `Preferences` page, create a new Resource of type `OracleDB` under `Default Resource Access Information`, as shown in the Figure 15–4.

Figure 15–4 Oracle Internet Directory Configuration



5. When creating the resource, in the `Resource Access Information` section, enter the database proxy user name, database proxy password, and the database value in the respective fields.

   A new key is created, and you can pass this key as a value for the `dbproxyConn` parameter. The number of proxy user connections and their access levels are set by the database administrator.

> **Note:** For creating a resource in OID, see Batch Loading in Section 17.3.3.2, "Populating Oracle Internet Directory". You can create a resource using sample LDIF available on OTN located at `http://www.oracle.com/technetwork/middleware/reports/overview/index.html`.

15.12.4 Configuration Settings in Reports Configuration Files

To use the database proxy feature, you must add and modify configuration settings in the Reports configuration files.

15.12.4.1 rwserver.conf

In the `rwserver.conf` file, the `dbproxy` key is set as follows:

```
<dbProxyConnKeys>
      <dbProxyKey name="key1" database="db1"/>
      <dbProxyKey name="key2" database="db2"/>
 </dbProxyConnKeys>
```

This configuration is optional. The `dbproxy` key is taken from the rwserver.conf file if the `dbproxyConn` parameter is not passed on the command line. The `dbproxykey` is

obtained from this configuration, based on the database that you specified in the `userid` command-line parameter.

15.12.4.2 rwservlet.properties

To use the database proxy feature through `rwservlet`, edit the `enabledbproxy` setting in the `rwservlet.properties` file as follows:

```
<enabledbproxy>yes</enabledbproxy>
```

By default, `enabledbproxy` is set to `no`. For `rwclient`, this configuration setting is not required.

15.13 Oracle Portal-Based Security for Backward Compatibility

Oracle Reports accomplishes both authentication and authorization through Oracle Platform Security Services.

For backward compatibility, 12*c* Release (12.2.1.3) supports:

- Security with OracleAS Portal 10*g* and Portal Classic 11*g*.
- Users and groups created in 10*g* Release 2 (10.1.2) identity management (IM).
- Non-secured mode of operation with Reports Server.

15.13.1 Security Features Provided by Oracle Portal

Oracle Portal provides a number of security features available to Oracle Reports Services that enable you to ensure that the appropriate users are getting important data in a secure fashion. With Oracle Portal security features in place, your users see only the data they're supposed to see.

Use Oracle 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

Oracle 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.

Oracle Portal is tightly integrated with Oracle Reports Services to create a robust and secure data publishing environment. Oracle Portal provides easy-to-use wizards for setting up Oracle 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 Oracle Portal repository. As an Oracle Portal user, you can then, optionally, publish registered RDFs and JSPs to an Oracle Portal page. As with all Oracle 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 Oracle Portal.

Access to Oracle 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 through Portal, you can still take advantage of the Oracle Reports Services' security features available in Oracle Portal to control user access to all of your reports.

When you expose a report as a portlet through Oracle Portal, Oracle Reports leverages OracleAS Single Sign-On, which eliminates the need for users to enter multiple log ins, 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, Oracle Portal automatically logs you into all registered portlet providers and subsystems.

For more information, refer to Chapter 16, "Deploying Reports in Oracle Portal".

15.14 Security Interfaces

The Security API of the Reports Software Development Kit (RSDK) enables you to integrate your own security model with the Reports Server. Oracle 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 Oracle Reports Services
- Register a security class with Oracle Reports Services
- Use the security class with Oracle Reports Services

16

Deploying Reports in Oracle Portal

The steps for deploying reports in Oracle Portal is the same in 12*c* Release (12.2.1.3) as in prior releases. However, the security mechanism underlying the deployment has changed. You can continue to use the security features in Oracle Portal from prior releases for backward compatibility, but you can now also choose to use the new Oracle Platform Security Services security mechanism. See Chapter 15, "Securing Oracle Reports Services".

This chapter describes how to use Oracle Portal to deploy your Oracle Reports Services reports. It includes the following sections:

- Creating Reports Users and Named Groups
- Registering Oracle Reports Components
- Publishing Your Report as a Portlet
- Troubleshooting Information

Before you deploy reports, both Oracle Portal and Oracle Reports Services must be installed and configured.

See also: The following resources for further information:

- Chapter 7, "Configuring Oracle Reports Services" for information on configuring Oracle Reports Services
- The *Oracle Portal Administrator's Guide* for information on configuring Oracle Portal
- The *Planning an Installation of Oracle Fusion Middleware* for information on installing both components
- The Oracle Fusion Middleware documentation CD
- The Oracle Technology Network, (`http://www.oracle.com/technetwork/index.html`)

16.1 Creating Reports Users and Named Groups

If you use the security features in Oracle Portal to control access to your reports, you must register all of your Reports users in Oracle Internet Directory and assign security privileges to all of them through Oracle Portal.

> **Note:** If you have a large user population already entered into an LDAP-compatible directory, you can use Oracle Internet Directory features to synchronize the directories and save yourself the effort of entering your users individually. You'll find information about Oracle Internet Directory's Directory Integration Server in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

In Oracle 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 Oracle Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by Oracle 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 using the environment switching functionality to dynamically set the environment for reports. Refer to Section 7.2.2, "Dynamic Environment Switching" for more information.

The next sections provide overview information on how to create users and groups in Oracle Portal. They include:

- Default Reports-Related Groups
- Creating Users and Groups

16.1.1 Default Reports-Related Groups

When you install Oracle 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 must assign appropriate privileges to these groups to enable group members to perform specific functions on reports through Oracle Portal. For example, for each report object that you want members of a group (for example, `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 (for example, `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.

Note: The RW_ groups are created automatically by configuring Oracle Portal, or you can create them manually. You can also run Web commands if they are in the IASADMINS group.

The following commands can be run by members of any group:

- getfile
- showmyjobs
- killmyjobs
- getjobid
- showjobid
- help

Only members of the RW_DEVELOPER group can run the following commands:

- showmap
- showenv
- showjobs
- parsequery

Members of the RW_ADMINISTRATOR group can run any command.

16.1.1.1 RW_BASIC_USER

Should the security check fail, members of the RW_BASIC_USER group 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.

16.1.1.2 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 members of this group 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.

16.1.1.3 RW_DEVELOPER

In addition to the privileges of the RW_POWER_USER group, the RW_DEVELOPER group can run the following Web commands that show the system environment:

- showmap
- showenv
- showjobs

- parsequery

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.

16.1.1.4 RW_ADMINISTRATOR

In addition to the privileges of the RW_DEVELOPER group, the RW_ADMINISTRATOR group has access to the administrator's functionality in the Oracle Reports Queue Manager, which means members of this group can manage the server queue, including rescheduling, deleting, reordering jobs in the server, and shutting down a server. Members of the RW_ADMINISTRATOR group can run any command. The RW_ADMINISTRATOR group 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 Oracle 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 Oracle 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 Reports Services.

16.1.2 Creating Users and Groups

Oracle Portal uses the Delegated Administration Service (DAS) interface to Oracle Internet Directory 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 Portal Administrator's Guide* available on the Oracle Fusion Middleware documentation CD.

When you create groups, you must assign appropriate privileges to them to enable group members to perform any desired functions on reports through Oracle Portal. For example, for each report object that you want members of a group (for example, 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 (for example, 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 (for example, JSMITH) rather than groups, but this approach is more difficult and time consuming to manage.

16.1.3 Portal Password in Credential Store

Oracle Reports 12*c* Release (12.2.1.3) uses credential store to store Portal password as a key. You can also use the credential store to configure database connection information for `jobstatusrepository` and `jobRepository` elements.

Portal password is stored in the reports credential map with key in the following syntax:

```
"portalpasswd_DomainName_InstanceName"
```

> **Note:** If you modify the Portal password, you must update the value of the key in the Reports credential store.

16.2 Registering Oracle Reports Components

Before you begin, you must have a sufficient level of privileges in Oracle Portal to access the portlets and complete the tasks required for setting access controls. In order to manage reports in Oracle 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 on joining privilege groups in Oracle Portal, refer to the *Oracle Portal Administrator's Guide*.

This section outlines the necessary steps to go about:

- Registering a Reports Server
- Registering a Report
- Registering a Printer
- Creating an Availability Calendar

To perform actions on existing Oracle Portal portlets, refer to:

- The Manage Portlet

16.2.1 Registering a Reports Server

Before you can define access controls for a Reports Server, you *must* register your server within Oracle Portal. Registration provides Oracle 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 16–1 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`*`, for example, rep_myserver1`

Table 16–1 (Cont.) Sample Values

Property	Sample Value
Oracle Reports Web Gateway URL for JSP reports	`http://myias.mycomp.com:7778/`
Oracle Reports Web Gateway URL for RDF reports	`http://myias.mycomp.com:7778/reports/rwservlet`
Availability Calendar	`COMCAL`

To register a Reports Server:

1. Log in as an administrator to Oracle 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 Oracle 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 Oracle Portal, for example, `MY_REPORTS_SERVER`. This name must follow the Oracle Portal rules for a valid component name; that is:
     * 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 Oracle 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 Oracle 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 server creation through OPMN.
   - (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 Oracle Reports Servlet (`rwservlet`) 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 7, "Configuring Oracle 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 Oracle 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 Oracle 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 13, "Configuring Destinations for Oracle 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: Section 16.2.4, "Creating 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:** Section 16.2.5, "The Manage Portlet" for more information on the fields and descriptions listed in the Manage portlet (that is, 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.

16.2.2 Registering a Report

Registering a report is a required step that enables 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, through a Oracle Portal parameter form, how users are to interact with the report.

User parameters are created in Oracle Reports Builder at the time of designing the report. You can assign values to these parameters when you run the report in Oracle Portal.

> **Note:** You can use the parameter settings available through Oracle Portal to duplicate or create a subset of the parameters defined in Oracle Reports Builder at design time. At runtime, the Reports Server disregards any parameters that you set in Oracle Portal not defined in Oracle Reports Builder at design time.

Registering a report within Oracle Portal creates an Oracle Portal component that can be deployed as a portlet through Portal. We recommend that you register only one instance of a report file in Oracle Portal. If you define multiple Oracle 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.

Table 16–2 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 Oracle 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 Definition File Access** link 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 Oracle Portal, for example, MY_REPORT. This name must follow the Oracle Portal rules for a valid component name; that is:
     * 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 Oracle 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 Oracle 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 standalone JSPs. For JSPs, you must define the name as *`virtual_path/reportname`*`.jsp`.

See Also:

- Appendix B, "Environment Variables" for more information on Oracle Reports-related environment variables.
- Chapter 7, "Configuring Oracle Reports Services" for more information on specifying the virtual path.

- (Optional) Enter a description for this report in the **Description** field.
- In the **Execute** field, select either **through servlet** or **as JSP**. The selection you make here will affect the choices that are available on the next wizard page.
  * **through servlet:** If you plan to run the report through Oracle Reports Servlet (`rwservlet`).
  * **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 Oracle Reports Servlet (`rwservlet`). 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:
    - **Types** specifies the destination types acceptable for this report. Select the destination types from among Cache, File, Mail, OraclePortal, OracleWireless, Printer, FTP, WebDAV, or custom destination types. If the server you associate with this report supports custom destination types, which you indicated when you registered the Reports Server in Oracle Portal, the types you indicated will display on this list.
    - **Formats** defines the acceptable output format(s) for this report. Choose from HTML, HTMLCSS, PDF, XML, RTF, Delimited, Spreadsheet, 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: Section 16.2.4, "Creating 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: Section 16.3, "Publishing Your Report as a Portlet" for more information on how to run your report from Oracle Portal.

19. Click the **Access** tab and select the **Publish as Portlet** box. This adds the report to the Portlet Repository, allowing you to add it to a page and publish your report as a portlet.
20. Click **Customize** to view the report's Runtime Parameter Form.

    Table 16–3 summarizes the options available on this page.

Table 16–3 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.

Table 16–3 (Cont.) Options on the Runtime Parameter Form

Option	Description
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.
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=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 Single Sign-On connection strings. Separate multiple strings with a comma (but no spaces). For more information on `SSOCONN`, refer to Section 17.3.3.1, "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.
Servlet Command Key	Optionally, enter the key from the `cgicmd.dat` key map file (see Section 18.14, "Using a Key Map 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 (for example, 90%) or in pixels (e.g, 700). If no value is specified, Oracle 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 (for example, 50%) or in pixels (e.g, 400). If no value is specified, Oracle 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` key map file. See Section 18.14, "Using a Key Map File". If you wish to enter multiple additional parameters, simply separate each entry with a space. For more information on the distribution XML file, see Chapter 20, "Creating Advanced Distributions".

16.2.3 Registering a Printer

It is not required that you register a printer within the security framework of Oracle 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 Oracle 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 Oracle Portal is meaningful for reports that you run through Oracle Portal as well as those you run through a standalone URL.

Once printers are registered within Oracle 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 Oracle 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 Oracle Portal user running a report request to select where output should be sent.

Table 16–4 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 Oracle 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 Oracle 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 Oracle Portal, for example, MY_PRINTER. This name must follow the Oracle Portal rules for a valid component name; that is:
     * 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 Oracle 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 Oracle 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 through 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: Section 16.2.4, "Creating 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: Section 16.2.5, "The Manage Portlet" for more information on the fields and descriptions listed in the Manage portlet (that is, Develop, Manage, and Access tabs).

12. Click **Close** to close this page and return to Oracle Portal's **Oracle Reports Security** page.

You have completed registering a printer with Oracle Portal. This registration is meaningful for reports that are run through Oracle Portal as well as those run outside of Oracle Portal.

16.2.4 Creating an Availability Calendar

Defining availability calendars is an optional step that enables 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.

16.2.4.1 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 Oracle 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 Oracle Portal, for example, MY_CALENDAR. This name must follow the Oracle Portal rules for a valid component name; that is:
     * 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 Oracle 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 Oracle 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** indicates that 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** indicates that 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** indicates that 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 (for example, always on the 25th through the 29th of the month) or by the particular day(s) (for example, always on Monday through Friday—which happen this month to fall on the 25th through the 29th).
    - **Weekly** indicates that the duration period restarts on a weekly basis between the days specified under **Duration**.
    - **Daily** indicates that 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 Oracle Portal's **Oracle Reports Security** page.

You can combine this calendar with other calendars or apply it "as is" to registered Oracle Reports Services components.

16.2.4.2 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 Oracle 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 Oracle Portal, for example, MY_ COMBINED_CALENDAR. This name must follow the Oracle Portal rules for a valid component name; that is:
     * 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 Oracle 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:** Section 16.2.5, "The Manage Portlet" for more information on the fields and descriptions listed in the Manage portlet (that is, Develop, Manage, and Access tabs).

18. Click **Close** to close this page and return to Oracle Portal's **Oracle Reports Security** page.

You can combine this calendar with other calendars or apply it "as is" to registered Oracle Reports Services components.

16.2.5 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 16–5 details the fields and descriptions listed in the **Develop** tab.

Table 16–5 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.
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 through 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.

Table 16–5 (Cont.) The Develop Tab

Field	Description
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 16–6 details the fields and descriptions listed in the **Manage** tab.

Table 16–6 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 16–7, Table 16–8, Table 16–9, Table 16–10, Table 16–11, Table 16–12, and Table 16–13 details the fields and descriptions listed in the **Access** tab.

Table 16–7 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 through Expose as Provider on the Access provider page (Manage tab).

Table 16–8 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 16–9 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 16–10 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 16–11 Cell Privilege Mode

Field	Description
Inherit Privileges from portlet (Not applicable to all portlets)	Select to allow the portlet access privileges to override cell access privileges. 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 16–12 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.

Table 16–12 (Cont.) Alter Access

Field	Description
Alter (Not applicable to all portlets)	Click to alter cell access privileges.

Table 16–13 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.

16.3 Publishing Your Report as a Portlet

After you have registered your Oracle Reports, you can expose your report in a portal by performing the following steps:

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. Alternatively, you can use the existing providers included, by default, with Oracle Portal.

   Refer to Section 16.3.1, "Creating a Provider for Your Reports".

2. Add the report as an item link[1] or as a portlet[2] to a page and optionally customize it. This 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. You can also distribute report output to Oracle Portal.

   Refer to:

   - Section 16.3.2, "Adding the Report Portlet to a Page"
   - Section 16.3.3, "Adding the Report as an Item Link to a Page"
   - Section 16.3.4, "Running Reports on Oracle Portal as an Item Link on a Nondefault Installation"
   - Section 16.3.5, "Distributing Report Output to Oracle Portal"

16.3.1 Creating a Provider for Your Reports

If you do not already have a provider defined to contain your reports, you must 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.

16.3.2 Adding the Report Portlet to a Page

After you have registered your report with Oracle Portal, you can publish it as a portlet on your portal page.

[1] An individual piece of content (text, hyperlink, image, and so on) that resides on a page in an item region.

[2] A reusable, pluggable Web component that typically displays portions of Web content.

Note: You must have enabled the **Publish as Portlet** box to ensure that you can publish your report as a portlet.

To publish a report as a 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: Section 16.3.3, "Adding the Report as an Item Link to a Page" for information on how to add the Oracle Reports item to a page.

   If you are editing an existing page, skip to the next step.
6. 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.

7. 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.
8. Click the name of your report portlet to add it to the **Selected Portlets** list.
9. Click **OK**.
10. Click **Customize** in the upper right corner of your report portlet.
11. Enter parameter values in the **Parameter** tab and, if desired, schedule the job to run automatically in the **Schedule** tab.
12. 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 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 through Customize. The user's value

will override the value set in the Customize page of the Reports Definition File Object component.

13. You can choose whether to make a report's parameters visible to users on 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 Develop tab for the report.

    b. Select **Visible to user** for each parameter you want to expose.

    Note: You can also set the default value of the parameter from this page.

    If the parameter you are exposing has a corresponding Oracle 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.

16.3.3 Adding the Report as an Item Link to a Page

You can add an Oracle Reports component to a page as an item link using the Oracle Reports item type. If you have installed Oracle Portal with the nondefault language setting, refer to Section 16.3.4, "Running Reports on Oracle Portal as an Item Link on a Nondefault Installation".

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.

   If you are editing an existing page, skip to the next step.
6. 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.
7. Select Oracle Reports in the **Hidden Item Types** list and click the right arrow (**>**) to move it to the **Visible Item Types** list. Alternatively, you can click **>>** to move all items in the **Hidden Item Types** list to the **Visible Item Types** list.
8. Click **OK**.

9. Select the Oracle Reports item type in the **Content Item Types** menu and click **Next** to display the Add Oracle Reports page.
10. Enter a **Display Name** that users of your portal will view when clicking your report.
11. Select from the list of available default Oracle Reports components.
12. Select **Display Parameter Form** if your report requires any user inputs before your report is displayed.
13. Select **Link That Displays Item In New Browser Window** to ensure that the report is viewed in a separate page.
14. Click **Finish**. The Oracle Reports item now displays as a link in your page.
15. Click the link to run the report and provide any parameters required, if **Display Parameter Form** is selected.

16.3.4 Running Reports on Oracle Portal as an Item Link on a Nondefault Installation

When you install Oracle Portal with a nondefault language setting, some entries required to publish a report as an item link on a portal page are not installed automatically. You must install the language of your choice by using the rwlang.sql script.

Thus, you must run the script `rwlang.sql` (`ORACLE_HOME/portal/admin/plsql/wwd/`) if both of the following are true:

- You have selected at least one language in addition to the default ("`US`") at the time of installing Oracle Portal.
- You want to publish a report as an item link in Oracle Portal.

> **Note:** This is a one time post-installation task and will ensure that you can publish a report as an item link on Oracle Portal.

To run the script:

1. Change the directory to `ORACLE_HOME/portal/admin/plsql/wwd/`.
2. Run `sqlplus`.
3. Log on to Oracle Portal using the portal schema.
4. This is the portal schema used to install Oracle Portal PL/SQL packages.
5. Run the rwlang.sql script with the following parameters:

```
@rwlang.sql language_list
```

where

`language_list` is the list of languages separated by commas.

For example, to install French and Japanese:

```
@rwlang.sql f,ja
```

Usage Note

- There should be no space before or after the comma (,) because sqlplus treats the language list as two parameters, instead of one parameter separated by a comma (,).

- The header of the rwlang.sql script contains the complete list of all language abbreviations. Edit the script file using any text editor to find out the various abbreviations.

16.3.5 Distributing Report Output to Oracle Portal

To distribute to the ORACLEPORTAL destination from Reports Server:

1. Create a distribution file (for example, distribution.xml) to distribute report output to Oracle Portal. For example:

```
<destinations>
  <destype id="customforPortal" name="oraclePortal">
    <property name="outputpage" value="Test_Reports"/>
    <property name="statuspage" value="result"/>
    <property name="pagegroup" value="test_reports"/>
    <property name="itemtitle" value="TestReport"/>
    <include src="report"/>
  </destype>
</destinations>
```

> **Note:** In this example, you must ensure that pagegroup test_reports exists in Oracle Portal or specify an existing pagegroup name.

2. Run a report with the following URL:

```
http://host:port/reports/rwservlet?server=reports_server
&report=report_name&destination=distribution_file
&distribute=yes&userid=user/password@db
```

3. In Oracle Portal, go to the page specified in the URL and make sure the report is visible.

16.4 Connecting to Oracle Portal

By default, Reports Server can only use Oracle Portal users to connect to Oracle 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 Oracle Portal schema, you must run the following script from SQL*Plus as an Oracle 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 (for example, repapp/repapp@orcl). The script then assigns the appropriate privileges to this new schema. You can then specify this connection string in the destination element in the Reports Server configuration file to connect to Oracle Portal.

16.5 Troubleshooting Information

This section contains information on the various steps that you can take to rectify issues that occur.

16.5.1 Resolving Reports-Portal Integration Error When Attempting Create Resource

In Oracle Portal, when configuring Oracle Reports Security settings for Reports Definition File Access, you may encounter an error when editing a report definition file, when you click **Run** or **Run as Portlet**.

```
500 Internal Server Error
Unexpected Error. Please contact Administrator
```

This error occurs when all of the following conditions are true:

- Running in an Interop deployment (which allows for a mixed 9.0.2/9.0.4 environment), with 9.0.4 MT (mid-tier), 9.0.4 IM (Identity Management), and 9.0.2 MR (metadata repository) configured to run together.
- Running Oracle Reports within Oracle Portal, using the SSOCONN parameter.
- The connection resource specified in the SSOCONN parameter has not been created in the Oracle Internet Directory server.

To implement the workaround, perform the following steps:

1. In the 9.0.4 IM *ORACLE_HOME*, open the following file in a text editor:

   ```
   ORACLE_HOME/Apache/Apache/conf/mod_osso.conf
   ```

2. Add the following flag:

   ```
   OssoRedirectByForm on
   ```

   For example:

   ```
   <IfModule mod_osso.c>
   OssoIpCheck off
   OssoIdleTimeout off
   OssoConfigFile
   /private1/iasinst/install_set1/904infra/Apache/Apache/conf/osso/osso.conf
   OssoRedirectByForm on
   </IfModule>
   ```

17

Configuring and Administering Oracle Single Sign-On

Oracle Single Sign-On (SSO) 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 in to an environment such as Oracle 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 Oracle Portal. Thus, logging in once provides them access to all of the resources and data sources they require to run their reports.

Oracle Reports Services applications can run in a Single Sign-on environment using Oracle Access Manager 11*g* (OAM) and Oracle Internet Directory (OID) to eliminate the need for additional or different logins to access many applications during the same user session.

Oracle Reports Services applications in Oracle FMW 12*c* Release (12.2.1.3) can now use one of the following authentication servers in the Single Sign-On mode:

- Oracle Access Manager (OAM) 11*g*

The user can choose to authenticate their Reports application using the authentication servers. It is required that these authentication servers are configured to use Oracle Internet directory as the backend Identity Store. Authentication servers are designed to work in Web environments where multiple Web-based applications are accessible from a browser. Without an authentication server, each user must maintain a separate identity and password for each application they access. Maintaining multiple accounts and passwords for each user is unsecure and expensive.

Because Oracle Reports Services provides a flexible approach to security, you can implement many variations of this configuration. 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 Oracle 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 Single Sign-On with Oracle Reports Services.

- Postinstallation Configuration
- Oracle Forms Services Security Considerations

17.1 Prerequisites

Single Sign-On (SSO) 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 Single Sign-On with Oracle Reports Services. For more information, refer to Chapter 15, "Securing Oracle Reports Services".

With Single Sign-On, your administrator establishes a user identity for each user. The administrator does this in Oracle Internet Directory. See *Oracle Fusion Middleware Administrator's guide for 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. 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 Oracle Internet Directory, see *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* and the *Administrator's Guide for Oracle HTTP Server*.

17.2 Configuring Single Sign-On

To run a report, you must login with a valid SSO userid and password. The Oracle Internet Directory instance installed with Oracle Fusion Middleware 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 Section 17.3, "Administering Single Sign-On". For information on how to add users to Oracle Internet Directory, refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*. In addition, for each Oracle Fusion Middleware installation, the Reports Server instances connect to Oracle Internet Directory as an application entity that is unique to the Oracle Fusion Middleware installation. For more information on this behavior, refer to Section 17.3.4, "Connecting to Oracle Internet Directory".

If a user is not already logged in to Single Sign-On, they are prompted to log in when they attempt to run a report to the Reports Server through `rwservlet`. If the user parameters for a report include `SSOCONN`, OracleAS Single Sign-On server will search for the user's data source credentials in Oracle Internet Directory. If none are found, then OracleAS Single Sign-On server prompts the user to create a new resource. For more information on `rwservlet`, refer to Section A.2.5, "rwservlet". For more information on `SSOCONN`, refer to Section 17.3.3.1, "SSOCONN". In case of OAM server, if the user's data source credentials do not exist in Oracle Internet Directory, then Oracle Reports raises a "key does not exist" error message. You must create a new resource in Oracle Internet Directory using the LDIF samples located on OTN at `http://www.oracle.com/technetwork/middleware/reports/overview/index.html`or see Section 17.3.3.2.2, "Batch Loading". See *Oracle Fusion Middleware Administrator's guide for Oracle Internet Directory*.

The Reports Server is also configured to operate with Oracle Portal by default if Oracle Portal is configured. 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 Single Sign-On mode by logging in to the portal.

You can also optionally define access controls for resources associated with the Reports Server (for example, reports, printers, Reports Servers, and calendars) in Oracle 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 Oracle 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 Oracle Portal.

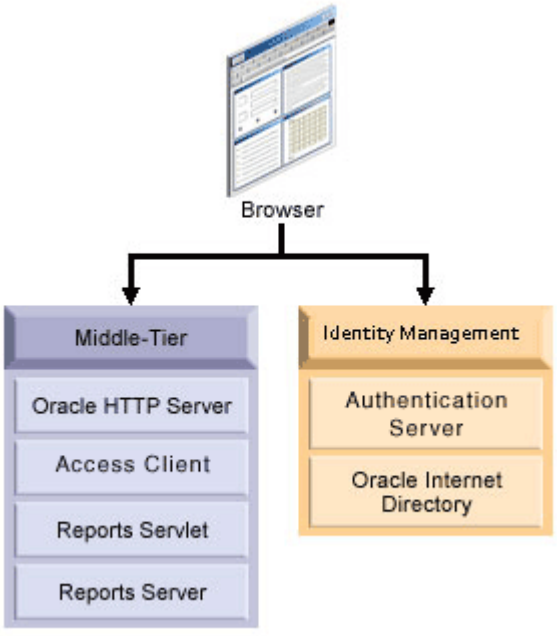
> **See Also:** Chapter 16, "Deploying Reports in Oracle Portal" for more information about the integration between Oracle Portal and Oracle Reports Services.

Note: In case of OSSO server, it is recommended that you use Single Sign-on to hide authid in URLs. For more information see, Section 7.3.1.1.18, "allowauthid".

17.2.1 Single Sign-On Components used by Oracle Reports

Figure 17–1 provides an overview of the Single Sign-On component architecture.

Figure 17–1 Single Sign-On 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.

> **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 *Administrator's Guide for Oracle HTTP Server*), `rwservlet` automatically detects the SSL port number.

- **Reports Servlet**

  Oracle Reports Servlet (`rwservlet`) is a component of Oracle Reports Services that runs inside Oracle WebLogic Server. When a report request comes to the Oracle HTTP Server, Oracle Reports Servlet (`rwservlet`) passes the job request to Reports Server.

- **Reports Server**

  Reports Server (`rwserver`) processes client requests, which includes ushering them through authentication and authorization checking, scheduling, caching, and distribution.

- **Authentication Server**

  - It verifies login credentials by looking them up in Oracle Internet Directory.
  - Oracle Access Manager (OAM server) - It is an Oracle FMW 11*g* authentication server that provides a full range of security functions that include Web single sign-on, authentication and authorization. When running Reports Services, it uses Oracle Internet Directory as the Identity Store. Oracle Access Manager can use either `mod_osso` or `webgate` as the access client configured with Oracle HTTP Server.

- **Access Client**

  - `webgate` - WebGate provides single sign-on support. It intercepts incoming HTTP requests and forwards them to the Access Server for authentication. Oracle Forms Services and Oracle Reports Services can use `webgate` as an access client with the authentication server.

- **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 Oracle Reports Services, it checks Oracle Internet Directory for user and group privilege information. It also retrieves data source connection information from Oracle Internet Directory.

- **Oracle Delegated Administration Services**

  The Delegated Administration Service provides a comprehensive interface for making updates to Oracle Internet Directory. Oracle 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 Oracle Internet Directory.

For more information, refer to Chapter 17, "Configuring and Administering Oracle Single Sign-On".

17.3 Administering Single Sign-On

This section describes some of the administrative tasks you may need to perform as you maintain security for Oracle Reports Services.

- Enabling and Disabling Single Sign-On
- Enabling and Disabling Reports Server Security
- Enabling and Disabling Data Source Security
- Connecting to Oracle Internet Directory

17.3.1 Enabling and Disabling Single Sign-On

To take advantage of Single Sign-on out-of-the-box, the `SINGLESIGNON` parameter in the Oracle Reports Servlet (`rwservlet`) configuration file (`rwservlet.properties`) is set to `YES`, which specifies that you will use OracleAS Single Sign-On to authenticate users. Oracle considers this to be the normal security deployment model and you should set `<singlesignon>no</singlesignon>` only if you plan to run in a completely custom security configuration.

When your Reports application is authenticated with OAM 11*g* server, even if the Single Sign-On parameter is set to `No`, the OAM authentication page is displayed and not the Reports authentication page.

To enable or disable OracleAS Single Sign-On, see Section 6.3.6, "Enabling and Disabling Single Sign-On".

17.3.2 Enabling and Disabling Reports Server Security

You can enable JPS-based security, including JAZN-XML authorization. See Chapter 15, "Securing Oracle Reports Services".

To enable or disable security, see Section 6.3.1, "Enabling and Disabling Security and Changing Security Mechanism used".

During Oracle Fusion Middleware installation, you are asked to select an identity store, a policy store, and a credential store. By default, these are file-based stores. After installation, you can change either of these to LDAP-based stores, such as Oracle Internet Directory. See the "Understanding Identities, Policies, Credentials, Keys, Certificates, and Auditing" chapter in *Securing Applications with Oracle Platform Security Services*.

17.3.3 Enabling and Disabling Data Source Security

To enable data source security through Single Sign-On, you must do the following:

- Include `SSOCONN` in the URL that launches the report.
- Populate Oracle Internet Directory with data source connection information using one of three methods.

If you wish to implement data source security through Single Sign-On for your own pluggable data sources, you must perform the following additional task:

- Add a new resource type to Oracle Internet Directory.

The following sections explain how to perform these operations.

17.3.3.1 SSOCONN

To enable data source security through Single Sign-On, the URL must contain or reference (that is, through the key map file) an Single Sign-On parameter (SSOCONN) with a value of the form:

key_name/data_source_type/conn_string_parameter

key_name maps to a string stored in Oracle Internet Directory that provides the necessary information to connect to the database.

The mapping of the *key_name* functions differently depending on the authentication server used in the Single Sign-On environment:

Mapping when Oracle Access Manager is used as the authentication server

In this case, when Oracle Reports encounters a *key_name*, it checks to see if the current user has a corresponding key stored in Oracle Internet Directory. If yes, 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 and uses that key. Otherwise, Oracle Reports raises a "key does not exist" error message.

> **See Also:** Section 17.3.3.2, "Populating Oracle Internet Directory" for more information about populating Oracle Internet Directory with resources.

data_source_type is the kind of data source to which you are connecting, to identify the format in the string associated with *key_name*. The *data_source_type* value must be a valid resource type stored in Oracle Internet Directory. Oracle Reports provides default resource types for the following:

- Oracle database (OracleDB)
- JDBC PDS (JDBCPDS)

You can also create additional resource types in Oracle Internet Directory for your own pluggable data sources.

conn_string_parameter 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 through the USERID parameter and uses it to connect to the specified Oracle database. Similarly, for JDBCPDS, P_JDBCPDS is used. If you have your own custom pluggable data sources, you must define your own user parameter for passing the connection string to Oracle Reports and specify it as *conn_string_parameter* for SSOCONN.

> **See Also:** Section A.8.15, "SSOCONN"

17.3.3.1.1 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
```

17.3.3.1.2 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: Section 14.1, "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`.

  Performing any of these actions with `SSOCONN` in the command line results in an error.

17.3.3.2 Populating Oracle Internet Directory

For data source security to function with Single Sign-On, you must store the data connection information for each user in 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:

- User Prompt
- Batch Loading

17.3.3.2.1 User Prompt If you prefer to have users enter their own connection string information, you do not have to prepopulate Oracle Internet Directory with data source connection information at all.

In case of OAM as the authentication server, if you use `SSOCONN` when launching the report but Oracle Internet Directory does not already contain a connection string for the key, then Oracle Reports raises a "key does not exist" error message. You must create a resource using the sample LDIF available on OTN located at `http://www.oracle.com/technetwork/middleware/reports/overview/index.html`. For more information about creating a resource in OID, see Section 17.3.3.2.2, "Batch Loading".

Note: Because of this feature, many users can use the same report URL even if they all use different data source connection strings.

17.3.3.2.2 Batch Loading Resources for Oracle Reports Services are created in 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]
```

Before You Begin You must create `orclownerguid=guid` in the 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.

[1] dc=us,dc=oracle,dc=com is merely an example in this instance. You would normally enter your own values for these items.

If you seeded users into Oracle Internet Directory with an LDIF file, then, before following the steps in Batch Loading Resources, you must complete the following steps:

1. Get the users' GUIDs.

   Depending on how your users are created in 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:\Oracle\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, *OTN*\scripts\createuser.ldif by replacing the place holder with real values.

   b. Load `createuser.ldif` using `ldapadd`. For example:

   ```
   D:\Oracle\BIN>ldapadd -D cn=orcladmin -w welcome1
   -h host_name -p port_num -f createuser.ldif
   ```

   Note: In the sub-step a, OTN referred to, is located at the Oracle Reports Samples http://www.oracle.com/technetwork/middleware/reports/downloads/index.html.

3. Once you have created `orclownerguid=guid`, proceed to Batch Loading Resources.

   Note: You can now find sample LDIF in the Oracle Reports page on OTN located at http://www.oracle.com/technetwork/middleware/reports/overview/index.html.

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, `C:\Samples\scripts\createresource.ldif` by replacing the place holder with real values.

   b. Load `createresource.ldif` using `ldapadd`. For example:

   ```
   D:\Oracle\BIN>ldapadd -D cn=orcladmin -w orcladmin's_password -h host_name
   -p port_num -f createresource.ldif
   ```

Note: In the sub-step a, the Samples referred to, should be downloaded from OTN location http://www.oracle.com/technetwork/middleware/reports/downloads/index.html and then be placed at local location *C:\Samples*.

17.3.4 Connecting to Oracle Internet Directory

As described in Chapter 15, "Securing Oracle Reports Services", Oracle Reports Services must connect to Oracle Internet Directory to verify user privileges and obtain existing data source connection information. In connecting to Oracle Internet Directory, you must consider:

- Choosing the Connecting Entity for Oracle Internet Directory
- Choosing the Oracle Internet Directory Instance

17.3.4.1 Choosing the Connecting Entity for Oracle Internet Directory

When Oracle Reports Services connects to Oracle Internet Directory, it does so as an application entity. By default, each Oracle Reports Services application entity is unique to its Oracle Fusion Middleware installation. Every Reports Server started from the same Oracle Fusion Middleware installation (that is, *ORACLE_HOME*) uses the same application entity to connect to Oracle Internet Directory. This setup ensures that each Reports Server can only access information in Oracle Internet Directory that is relevant to its instance of Oracle Fusion Middleware.

For example, suppose you have two instances of Oracle Fusion Middleware, one for your Finance group and one for your Human Resources group. A Reports Server from the Finance group's Oracle Fusion Middleware instance would be prevented from accessing information relevant only to the Human Resources group, and vice versa. Thus, information stored in Oracle Internet Directory is more secure by default.

In previous releases of Oracle Reports Services, all Reports Servers connected to 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 Oracle Internet Directory.

To revert to the less restrictive security mode, refer to the Oracle Reports Services chapter of the *Oracle Fusion Middleware Release Notes*.

17.3.4.2 Choosing the Oracle Internet Directory Instance

By default, the Reports Server is configured to use the Oracle Internet Directory instance installed with Oracle Fusion Middleware. If you are building your system anew, this arrangement is fine. However, if 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 must be done as part of a complete change of your Oracle Fusion Middleware middle tier. For more information about this process, refer to the chapter on reconfiguring Application Server instances in the *Administering Oracle Fusion Middleware*.

17.4 Choosing the Connecting Entity for Oracle Internet Directory

You can merge several application entities so that the Reports Servers installed in separate ORACLE_HOMEs 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 representing the logical grouping of all report instances
changetype: modify
add: uniquemember
uniquemember: dn of the Reports Application Entity
```

where:

dn of the group representing the logical grouping of all report instances is `cn=Virtual Application Group, orclApplicationCommonName=reports_application_entity_name**, cn=Reports, cn=Products, cn=OracleContext`

dn of the Reports Application Entity is `orclApplicationCommonName=reports_application_entity_name**, cn=Reports, cn=Products, cn=OracleContext`

** *reports_application_entity_name* is in the format `reportsApp_hostname_GUID`. For example, `reportsApp_serv1.example.com_C7543D42A9E26726E034080020A46EE`

Example

Entry in the LDIF file:

```
dn: cn=Virtual Application Group,
  orclApplicationCommonName=reportsApp_Group.example.com_C7543D42A9E26726E0340
  80020A46EE2, cn=Reports, cn=Products, cn=OracleContext
changetype: modify
add: uniquemember
uniquemember:
  orclApplicationCommonName=reportsApp_serv1.example.com_A8654E53B0F37837F1451
  91131B57FF3, cn=Reports, cn=Products, cn=OracleContext
```

Corresponding `ldapmodify` command on the command line:

```
ldapmodify -D cn=orcladmin -w welcome1 -h reportsApp_Group.example.com -p 389 -f
mergeentity.ldif
```

Note: You can now find sample LDIF in the Oracle Reports page on OTN located at http://www.oracle.com/technetwork/middleware/reports/overview/index.html.

17.5 Postinstallation Configuration

This section describes some postinstallation steps. This section includes the following topic:

- Installing and Configuring Webgate with OAM

17.5.1 Installing and Configuring Webgate with OAM

For webgate to work with Oracle Access Manager 11*g*, you must install and configure webgate manually. For information about installing and configuring webgate as the access client, see Installing and Configuring Oracle HTTP Server 11g Webgate for OAM in *Oracle Fusion Middleware Installation Guide for Oracle Identity Managment*.

Postinstallation, you must register webgate with OAM 11*g* so that webgate can directly communicate with Oracle Access Manager 11*g* services. Registration with OAM can be done by creating OAM 11*g* agent by using either RREG tool or through OAM console.

For information about registering webgate as an agent by using either OAM console or RREG tool, see Register the New Webgate Agent in *Oracle Fusion Middleware Installation Guide for Oracle Identity Managment*.

Reports OAM Integration

17.5.1.1 Webgate

The `mod_osso` module is no longer included with Oracle HTTP Server. Oracle WebGate is the recommended replacement. WebGate is now installed with Oracle HTTP Server. For more details, see https://docs.oracle.com/middleware/1213/webtier/HSADM/whats_new.htm#HSADM1182

17.5.1.2 Webgate Configuration

Configure webgate in classic shiphome domain

Refer to https://docs.oracle.com/middleware/1213/core/WTINS/webgate.htm#WTINS870 for more details.

```
$MW_HOME/webgate/ohs/tools/deployWebGate/deployWebGateInstance.sh -w $DOMAIN_
HOME/config/fmwconfig/components/OHS/instances/ohs1 -oh $MW_HOME

setenv LD_LIBRARY_PATH $MW_HOME/lib

$MW_HOME/webgate/ohs/tools/setup/InstallTools/EditHttpConf -w  $DOMAIN_
HOME/config/fmwconfig/components/OHS/instances/ohs1 -oh $MW_HOME -o webgate.conf
```

17.5.1.3 Registering an OAM Agent Using the Console

For more details, see https://docs.oracle.com/cd/E28280_01/admin.1111/e15478/agents.htm#AIAAG219.

1. Open the OAM Administration console and Login using OAM credentials.

   `http://<OAM_HOST>:<OAM_PORT>/oamconsole`

2. Click on **New OAM 11g Webgate** link on the welcome page.
3. Provide the following information in the "Create OAM 11g Webgate" screen
   - **Name**: Any name to the webgate agent
   - **Base URL**: http://<OHS_HOST>:<OHS_PORT>
   - **Protected Resource list**: /reports/rwservlet/*
   - **Public Resource List**: / and /.../*

> **Note:** We are currently seeing some issues with the Webgate public resource list. So skip steps Protected Resource List and Public Resource List. This will flatly protect all the entries accessed through OHS Host/port.

4. Click **Apply** and, you will get a information box indicating the location of the artifacts generated.

Copy the output to OHS instance directory in Classic domain

- cp <OAM webgate artifacts> `$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/webgate/config`

  The following two files will be copied:

- cwallet.sso
- ObAccessClient.xml

Restart OHS

Restart the OHS using the following commands:

```
DOMAIN_HOME/bin/stopComponent.sh  ohs1
```

```
DOMAIN_HOME/bin/startComponent.sh  ohs1
```

To access OHS reports url and test use the following command:

```
http://ohsHost:ohsPort/reports/rwservlet/help
```

> **Note:** Instead of using OAM console you can also use RREG script to register a OAM agent. Refer to the links mentioned above.

17.6 Oracle Forms Services Security Considerations

The default configuration for Oracle Fusion Middleware Forms Services does not run in the Single Sign-On (SSO) mode. The default configuration for Oracle Reports Services does run in SSO mode.

Oracle Forms Services applications calling integrated Oracle Reports Services using the `RUN_REPORT_OBJECT` built-in procedure will not experience any problems when Oracle Forms Services is running in non-SSO mode and Oracle Reports Services is running in Single Sign-On mode as long as the Reports Server and the requested report are not registered in Oracle Portal.

Other Requirements:

- The property Reports Server must be set explicitly for all report objects in the Oracle Forms Services module.
- If a Reports Server other than the default is being used, that server must be started (using Oracle Enterprise Manager).
- The system variable `REPORTS_PATH` must be modified in the file *`$DOMAIN_HOME/reports/bin/reports.sh`* to reference the path of the reports to be run.
- The first time Reports Server is started, it creates a configuration file called *`rwserver.conf`* located in the *`${DOMAIN_`*

HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/ directory.

- The default status of Reports Server is secure. To change the Reports Server status to non-secure, modify *${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/rwserver.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 (using FMW scripts).
- If Oracle Forms Services is configured to run in Single Sign-On mode, then report requests are sent with the `authid` provided, based on the Single Sign-On user login.
- Protected reports and Reports Servers can be registered in Oracle Portal.

Table 17–1 lists the possible Forms/Reports combinations and expected results:

Table 17–1 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

Part IV

Sending Requests to the Server

Part IV 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 through XML customization files, and how to use database triggers to automatically invoke reports:

18

Running Report Requests

This chapter discusses various ways to send report requests to the Reports Server. It includes the following sections:

18.1 The Reports URL Syntax

This section provides quick reference information on formulating a URL for publishing a report. It covers two deployment types:

The information is largely the same for both Windows and UNIX environments. Differences are noted.

18.1.1 Oracle Reports Servlet

The syntax for the URL to run a report through Oracle Reports Servlet (`rwservlet`) is:

`http://`*`web_server.domain_name:port/alias`*`/rwservlet?`*`parameters`*

Table 18–1 lists and describes the components of the URL.

Table 18–1 Components of a URL That Calls Oracle 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 Oracle 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 Oracle Reports Servlet (rwservlet) could look like this:

```
http://neptune.world.com:80/reports/rwservlet?keyname
```

where *keyname* refers to a command listed under a unique header (the key name) in the cgicmd.dat key map file. See Section 18.14, "Using a Key Map 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.

When you use Oracle Reports Servlet (rwservlet), you can also execute JSP report files if the JSP files contain paper layouts. When you run the report, specify Oracle Reports Servlet (rwservlet) 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 and values in Appendix A, "Command-Line Keywords".

18.1.2 JSP

The syntax for a JSP-based report URL is:

```
http://web_server.domain_name:port/alias/myreport.jsp?parameters
```

Table 18–2 lists and describes the components of the JSP-based report URL.

Table 18–2 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.

Table 18–2 (Cont.) Components of a JSP-based Report URL

Component	Description
`myreport.jsp`	The report `*.jsp` file that you want this URL to execute.
`?`	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 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 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 after 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.

When you use a JSP, you can also use Oracle Reports Servlet (`rwservlet`). When you run the report, specify Oracle Reports Servlet (`rwservlet`) 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&de
sformat=html
```

For more information about command line keywords, see Appendix A, "Command-Line Keywords".

18.2 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 component 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 Appendix A, "Command-Line Keywords" for more information about command line options.

- **A URL**

To run a report from a browser, use the URL syntax. Oracle Reports Servlet (`rwservlet`) converts the URL syntax into an `rwclient` command line request that is processed by Oracle 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.

- **Through Oracle Portal**

  The Oracle Portal component enables you to add a link to a report in an Oracle 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. Oracle Reports Services system administrators use Oracle Portal wizards to create the packaged procedure making it more convenient and secure to publish the report through the Web. Authorized users accessing the Oracle Portal page group simply click the link to run the report. System administrators can run the report directly from the wizard. See the *Oracle Portal online Help* for more information.

  Refer to Section 18.5, "Publishing a Report in Oracle Portal" for more information about how to publish your report as a portlet.

- **A packaged procedure**

  `SRW.RUN_REPORT` is a built-in procedure 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 a `rwclient` command.

  See Chapter 21, "Using Event-Driven Publishing". For a description of `SRW.RUN_REPORT`, refer to the *Oracle Reports online Help*.

- **A Web service**

  You can expose Oracle Reports Services as a Web service and then call it from any Web service aware environment (for example, a Java application).

  See Chapter 19, "Using the Oracle Reports Web Service".

18.3 Reports OHS Integration

In the below examples, the following are used:

- OHS name is 'ohs1'
- reports managed server host1 - host1.example.com
- reports managed server host2 - host2.example.com
- reports managed server port1 - 9001
- reports managed server port2 - 9002

Once OHS is created create a file like below with contents given below:

- `DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/moduleconf/reports_ohs.conf`
- Restart OHS

WLS_REPORTS in cluster

```
#mod_weblogic related entry
```

```
#<IfModule mod_weblogic.c>
  <Location /reports>
      SetHandler weblogic-handler
      WebLogicCluster    host1.example.com:9001,host2.example.com:9002
  </Location>
#</IfModule>
```

WLS_REPORTS not in cluster

```
#mod_weblogic related entry
#<IfModule mod_weblogic.c>
  <Location /reports>
      SetHandler weblogic-handler
      WebLogicHost host1.example.com
      WebLogicPort 9001
  </Location>
#</IfModule>
```

18.4 Deploying Your Reports

Once you have 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 (that is, REP, RDF, XML, or JSP report) and how to deploy a report with a Web layout (that is, 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 simple Parameter Form for a JSP-based Web report" in the *Oracle Reports User's Guide to Building Reports* manual.

The following table describes which method you can use to deploy your report, depending on the type of report.

Table 18–3 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 Section 18.4.3, "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.

Note: `rwrun` and `rwclient` execute the paper layout of your report. The report is processed and executed even though your JSP has only a paper layout. If your JSP has only a web layout but not a paper layout, running the JSP report using `rwrun` or `rwclient` obtains a blank output. If you have a JSP with paper layout, it is recommended that you save the JSP as RDF and then run the RDF using `rwrun` or `rwclient`.

18.4.1 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 Section 18.4.3, "Deploying a JSP Report to the Web and to Paper".

If your report depends on Java classes (for example, Barcode classes, a Web Service stub, and so on), 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 engine element in the server configuration file (`${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/rwserver.conf` for Standalone servers and `$DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/rwserver.conf` for In-process servers).

To deploy your paper report:

1. Transfer the report file (RDF, REP, XML, or JSP) and its associated files (for example, 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.

2. Ensure 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.

   Note: The report requests fail with a REP-110 error in case the paper layout JSP files are not available on the file system where report server is running. This error does not occur for JSP reports that contain a web layout. Therefore ensure that the paper layout JSP reports are available in the directory that is included in the REPORTS_PATH variable.

18.4.2 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/reports/rwservlet?server=server_name&report=
myreport.rdf&userid=username/password@my_db&desformat=pdf&destype=cache
```

In this example, your report displays in PDF format (`desformat=PDF`) in the browser.

For more information on running a report from the browser, refer to Section 18.6, "Specifying a Report Request from a Web Browser".

18.4.3 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 Java EE 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.

Note: The easiest way to deploy JSP reports is to copy them to the following directory:

```
$DOMAIN_HOME\servers\WLS_REPORTS\tmp\_WL_user\reports_release\dir_
name\war
```

The procedure described in this section for building your own EAR file and deploying it is only indicative; it is not comprehensive. For the detailed procedure, see *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.

About JSP reports with both paper and Web layouts

With Oracle 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 through the Oracle Reports engine.
- JSP reports with Web layouts are executed through the Java EE container.

If your report depends on Java classes (for example, Barcode classes, a Web Service stub, and so on), 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 engine element in the server configuration file (`${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/rwserver.conf` for Standalone servers and `$DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/rwserver.conf` for In-process servers.)

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 Java EE container `classpath`. For more information, refer to the *Oracle Containers for Java EE* documentation.

Note: For an example on building a report with a paper and Web layout, see "Building a Report with a Barcode" in the *Oracle Reports User's Guide to 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 Java EE application. To deploy your JSP report with a paper and Web layout, you can create a new Oracle Reports Java EE application. You can create this application in an existing instance or a new instance of Oracle WebLogic Server.

18.4.3.1 Creating a New Java EE Application

In this section, you will create a new Java EE application for Oracle Reports. You will create a Web application archive (WAR file) that will contain the application information, then deploy it as an Enterprise archive (EAR file). To create a new Java EE 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 must make a few modifications to the application, as well as to your JSP report.

To create a Java EE application:

Note: If you are not familiar with creating a Java EE application, refer to Sun Microsystem's Web site (`http://java.sun.com/javaee`). For more information on using Oracle JDeveloper, refer to the *Oracle JDeveloper online Help.*

1. Before you create your EAR file, ensure that 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 Fusion Middleware, you can find the tag library here:

   DOMAIN_HOME`/servers/WLS_REPORTS/tmp/_WL_user/reports_`
   version`/`*random_string*`/war/WEB-INF/lib/reports_tld.jar`

2. Ensure that your JSP-based Web report points to the location of the JSP tag library for Oracle Reports. Otherwise, the report will not run.

   To point to the location of the JSP tag library, include the `taglib` directive in the JSP file:

   ```
   <%@ taglib uri="/WEB-INF/lib/reports_tld.jar" prefix="rw" %>
   ```

3. Create a new EAR file, either manually or using a tool such as Oracle JDeveloper. Ensure you create the WAR file according to the appropriate Java EE 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: On Oracle Fusion Middleware, the `web.xml` file is located here:

DOMAIN_HOME`/servers/WLS_REPORTS/tmp/_WL_user/reports_`*version*`/`*random_string*`/war/WEB-INF/lib/reports_tld.jar`

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>1</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 starting with `/rwservlet` to the Oracle Reports Servlet you've defined.

Note: You can change the Oracle Reports 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.

18.4.3.2 Deploying Java EE Application Using WebLogic Server

After you have created the WAR and EAR files, you can deploy them to the Oracle Fusion Middleware, which will serve the application to the Web. You can deploy these files using Oracle Enterprise Manager using either an existing WebLogic Server instance or a new WebLogic Server instance. For more information about deploying Java EE application in Oracle WebLogic Server, see *Deploying Applications to Oracle WebLogic Server*.

18.4.4 Running a JSP-Based Web Report from a Browser

If your JSP report is a Web report, you can 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 this location.

- Re-create the WAR file with the modified JSP-based Web report, then redeploy the application. For more information, refer to Section 18.4.3.1, "Creating a New Java EE Application".

For more information on running a report from a browser, refer to Section 18.6, "Specifying a Report Request from a Web Browser".

18.4.5 Running a JSP report with a Paper Layout

If your JSP report has a paper layout, you can 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
```

In this example, your report displays as a PDF (`desformat=PDF`) in the browser.

For more information on running a report from a browser, refer to Section 18.6, "Specifying a Report Request from a Web Browser".

18.4.6 Running with the WE8MSWIN1252 Character Set on UNIX

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 `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin` directory.

Note: This mapping is required for Oracle Reports Builder, Reports Converter in non-batch mode (`BATCH=NO`), Reports Server and Reports Runtime with `REPORTS_DEFAULT_DISPLAY=NO`. Reports Server and Reports Runtime uses `REPORTS_DEFAULT_DISPLAY` to determine the fonts needed.

Example1

`Tk2Motif*fontMapCs: ISO8859-1=WE8MSWIN1252` (if there are `ISO8859-1` fonts installed).

18.5 Publishing a Report in Oracle Portal

One of the best ways to publish your report is through the declarative, secure interface of Oracle Portal. To publish a report in Oracle Portal, refer to Chapter 16, "Deploying Reports in Oracle Portal". Specifically, you must first register your Oracle Reports components in Oracle Portal (see Section 16.2, "Registering Oracle Reports Components"), then expose your report in a portal (see Section 16.3, "Publishing Your Report as a Portlet").

Note: When you use features like Oracle Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by Oracle 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 using the environment switching functionality to dynamically set the environment for reports. Refer to Section 7.2.2, "Dynamic Environment Switching" for more information.

18.6 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 keywords, then refer to Appendix A, "Command-Line Keywords" for a list of valid rwservlet command line keywords.

- Simplified URL request using key mapping, for example:

  ```
  http://your_webserver.domain_name:port/alias/rwservlet?key1
  ```

18.7 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 (for example, rwurl) in the server configuration file that is associated with a URL engine.

Note: For information on activating the URL engine, refer to Section 7.6, "Configuring the URL Engine".

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=rwurl+urlParameter=
"http://www.oracle.com"+destype=mail+desname=foo@bar.com+desformat=htmlcss
```

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=cache+urlParameter=
"http%3A%2F%2Flocalhost%2Ffoo.jsp%3Fuserid%3Dscott%2Ftiger@oraDB%3Fserver%3DreportsServer"
```

> **Note:** If the URL has special characters, they must be encoded as per the `x-www-form-urlencoded` format.

18.8 Running Reports Through a Web Service

In many cases, reports are integrated components of some larger application rather than a standalone application themselves. Hence, it can be useful to generate report requests from within an application. We accomplish this goal by exposing Oracle Reports Services as a Web service. This Web service may then be called from within any Web service-aware environment (for example, 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 (for example, Oracle JDeveloper) and add a button that invokes Oracle Reports Services to generate the PDF file.

> **See Also:** Chapter 19, "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.

18.9 Calling Oracle Reports from Oracle Forms Services

The tight product integration between Oracle Reports and Oracle Forms Services enables you to pass blocks of data between the two 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 through triggers and lexical parameters.

Oracle Forms Services uses the shared Java Virtual Machine (JVM) controller for all report requests, reducing memory consumption.

> **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.

A typical integration between Oracle Forms Services and Oracle Reports is an application that provides a form to fill in data, which is used to generate a report. The steps that occur during this process are similar to the following example:

- The end user enters values in a form. Some or all of these values are parameter inputs to the associated report.
- The end user clicks a button, which generates the report as follows:
  - Oracle Forms Services populates all the parameters to execute the report.
  - Oracle Forms Services calls the `RUN_REPORT_OBJECT` built-in to send a request to Oracle Reports.

- Oracle Reports returns the job id, and Oracle Forms Services queries on the status of this job id.
- When the job status is `FINISHED`, Oracle Forms Services calls the `WEB.SHOW_DOCUMENT` built-in to submit a request to open the report output.
- The `WEB.SHOW_DOCUMENT` built-in opens the following URL in the browser:

  `http://host:port/reports/rwservlet/getjobidn?server=server_name`

  Note: For secure mode, the URL will also include `authid=authid`

  For additional information on calling a report from an Oracle Forms Services application, refer to the *Integrating Oracle Forms 10g and Oracle Reports 10g* white paper on OTN (http://www.oracle.com/technetwork/developer-tools/forms/documentation/techlisting10g-085500.html).

18.9.1 Communication Between Reports and Forms Installed on Different Instances

Oracle Reports communicates with Forms. If Forms and Reports are configured on different Oracle Instances, you must complete the following steps in Forms to facilitate communication with Reports Servers.

1. Create a reports tools component targeted to the machine where forms is running.
2. Set this environment variable in the forms JVM `COMPONENT_CONFIG_PATH=${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>`

If Forms and Reports are configured on different OHS Servers, you must route the web requests from OHS instance where Forms is configured to the Reports Servlet as follows:

1. Copy the reports_ohs.conf file as follows:

   ```
   cp $ORACLE_HOME/reports/conf/reports_ohs.conf $FORMS_ORACLE_INSTANCE/config/OHS/ohs1/moduleconf/.
   ```

2. Replace the macros `$$managedserverhost$$` and `$$managaerserverport$$` with host and port of the `WLS_REPORTS` managed server.
3. Restart the OHS that is running in `FORMS_ORACLE_INSTANCE`.

18.9.2 Generating Random and Non-Sequential Job IDs

Oracle Reports allows you to generate random and non-sequential job IDs to make it impossible to predict the job ID for a particular job. To generate random and non-sequential job IDs for in-process servers, you must pass `"-Djobid=random"` via JVM options to Oracle WebLogic Server.

For standalone servers, you can generate random and non-sequential job IDs by passing the `"-Djobid=random"` via JVM options in the command line or by setting the `REPORTS_JVM_OPTIONS` variable. See Section B.1.55, "REPORTS_JVM_OPTIONS".

This prevents malicious users from viewing non-secure report output by typing the job id in a URL.

18.10 Running Reports Using Oracle BPEL Process Manager

Oracle Reports is integrated with the Oracle Service-Oriented Architecture (SOA) suite, which includes Oracle BPEL Process Manager to automate and monitor reporting requirements.

You can submit Oracle Reports jobs from the Oracle BPEL Process Manager business process, get the status of report execution, and also invoke an Oracle BPEL Process Manager business process from your reports. This makes it easy to include reporting requirements in the business process; for example, submit a report request when an order gets approved.

See Section 7.10, "Configuring Oracle Reports to Communicate with Oracle BPEL Process Manager", and also *Developing SOA Applications with Oracle SOA Suite*.

18.11 Scheduling Reports to Run Automatically

You can use the server to run reports automatically from Reports Queue Manager (`rwrqm` on Windows, or `rwrqv.sh` on Solaris), Oracle Portal, or with the `SCHEDULE` command line keyword. 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 Oracle Portal page, then you can schedule the report request to run automatically and push the resulting report output to specified pages. Refer to the *Oracle Portal online Help* for more information.

The `SCHEDULE` command line keyword is available for use with the `rwclient` and `rwservlet` commands. See Section A.8.3, "SCHEDULE" for more information.

18.12 Additional Parameters

When you send a request to the Reports Server through `rwcgi`, the following additional parameters, the values of which you cannot change, are implicitly passed along with your request:

Table 18–4 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 Oracle Reports Servlet (`rwservlet`) is running.
`SERVER_PORT`	The port number of the server on which Oracle Reports Servlet (`rwservlet`) 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.

18.13 Reusing Report Output from Cache

When you run a report, a copy of the report output is saved in the Oracle Reports Services cache. Subsequently, if an identical report is run (that is, with the 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 need not 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", then Reports Server will try to find a job in the Finished Jobs Queue than was successfully completed within *`n`* minutes. If Reports Server finds such a job, then the new job request "A" will return the output of job "B".

  Note: Refer to Section A.8.22, "TOLERANCE" for more information.

Oracle 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.

18.13.1 Usage Note

You can set the cache size through Reports Queue Manager (`rwrqm` on Windows, or `rwrqv.sh` on Solaris) or through the `cache` element in the server configuration file (*`rwserver`*`.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 Chapter 7, "Configuring Oracle Reports Services".

18.14 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. This section provides the following information:

- Understanding Key Mapping
- Enabling Key Mapping
- Adding Key Mapping Entries to a Key Map File
- Using a Key with Non-JSP Reports
- Using a Key with a Report Run as a JSP

18.14.1 Understanding Key Mapping

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. Oracle Reports Servlet (`rwservlet`) sends the key value to the key map file (`cgicmd.dat`), which in turn returns the command associated with the specified key to `rwservlet` 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: See Section A.5.14, "CMDKEY" for more information.

18.14.2 Enabling Key Mapping

Key mapping is enabled when a valid file with the standard file name, `cgicmd.dat`, is present in the default location: `$DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/cgicmd.dat` directory on the Web server machine (on either Windows or UNIX).

18.14.3 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:

   ```
   $DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_
   <version>/configuration/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 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 Section A.2.1, "rwclient" for more information.

3. Add or update the hyperlinks on your Web page.

See Section 18.6, "Specifying a Report Request from a Web Browser".

18.14.4 Using a Key with Non-JSP Reports

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
```

The key name is case-sensitive; that is, it must exactly match the case specified in the key mapping file (`cgicmd.dat`).

The following example shows a key mapping for a restricted run with a parameter form.

The URL might be:

```
http://web_server.domain_name:port/reports/rwservlet?run_report&par1&par2&parN
```

The key mapping file (`cgicmd.dat`) might contain:

```
run_report: report=myreport deptno=%1 myparam=%2 %*
```

This generates the equivalent of the following command line request:

```
rwclient report=myreport deptno=par1 myparam=par2 parN
```

18.14.5 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. The `cmdkey` value (*keyname*) is case-sensitive; that is, it must exactly match the case specified in the `cgicmd.dat` file. For example, your JSP URL might look like this:

```
http://…/myreport.jsp?cmdkey=keyname
```

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 Note

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 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. In this example, the URL becomes:

```
http://neptune.world.com:80/jsp/myreport.jsp?userid=scott/tiger@hrdb
  &DEPTNO=10&MYPARAM=test
```

19

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:

- Includes 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 need not 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:

19.1 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 through the event-driven publishing API.

- `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, requires 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, and is required to be exposed and to function as a Web service. This functionality enables any application developer to include Oracle Reports in their application.

19.2 Getting Started

This section outlines the steps necessary for:

- Invoking the RWWebService Servlet
- Viewing the WSDL

19.2.1 Invoking the RWWebService Servlet

To invoke the `RWWebService` servlet:

1. Start a Oracle WebLogic Server instance, where the Oracle Reports instance resides.
2. Enter the following URL in the address field of your browser:
   `http://`*`yourwebserver: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.

19.2.2 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 **WSDL** link on the `RWWebService` Web page to view the Oracle Reports Web service's WSDL document.

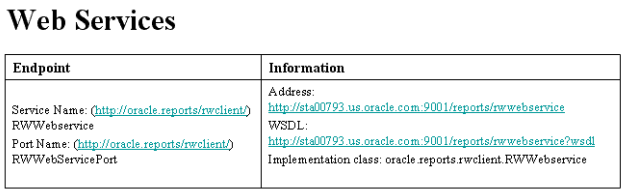
Note: Use Internet Explorer to view the WSDL XML output.

---

2. The last entry in the WSDL is the service description and contains the location of the WebService:

```
<soap:address location="http://yourwebserver:8888/reports/rwwebservice" />
```

Figure 19–1 Viewing the WSDL



Ensure that the URL and port number defined, `http://`*`yourwebserver`*`:`*`port`*`/reports/rwwebservice`, is correct.

> **Note:** The hostname specified should be the hostname where the Oracle WebLogic Server instance is running and not where the Reports Server is running.

Oracle Reports WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Oracle
JAX-WS 2.1.4. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Oracle
JAX-WS 2.1.4. -->
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://oracle.reports/rwclient/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://oracle.reports/rwclient/" name="RWWebService">
<types>
<xsd:schema>
<xsd:import namespace="http://oracle.reports/rwclient/"
schemaLocation="http://localhost:8888/reports/rwwebservice?xsd=1"/>
</xsd:schema>
</types>
<message name="getServerInfo">
<part name="parameters" element="tns:getServerInfo"/>
</message>
<message name="getServerInfoResponse">
<part name="parameters" element="tns:getServerInfoResponse"/>
</message>
<message name="getJobInfo">
<part name="parameters" element="tns:getJobInfo"/>
</message>
<message name="getJobInfoResponse">
<part name="parameters" element="tns:getJobInfoResponse"/>
</message>
<message name="getAPIVersion">
<part name="parameters" element="tns:getAPIVersion"/>
</message>
<message name="getAPIVersionResponse">
<part name="parameters" element="tns:getAPIVersionResponse"/>
</message>
<message name="killJob">
<part name="parameters" element="tns:killJob"/>
</message>
<message name="killJobResponse">
<part name="parameters" element="tns:killJobResponse"/>
</message>
```

```
<message name="runJob">
<part name="parameters" element="tns:runJob"/>
</message>
<message name="runJobResponse">
<part name="parameters" element="tns:runJobResponse"/>
</message>
<portType name="RWWebService">
<operation name="getServerInfo">
<input message="tns:getServerInfo"/>
<output message="tns:getServerInfoResponse"/>
</operation>
<operation name="getJobInfo">
<input message="tns:getJobInfo"/>
<output message="tns:getJobInfoResponse"/>
</operation>
<operation name="getAPIVersion">
<input message="tns:getAPIVersion"/>
<output message="tns:getAPIVersionResponse"/>
</operation>
<operation name="killJob">
<input message="tns:killJob"/>
<output message="tns:killJobResponse"/>
</operation>
<operation name="runJob">
<input message="tns:runJob"/>
<output message="tns:runJobResponse"/>
</operation>
</portType>
<binding name="RWWebServicePortBinding" type="tns:RWWebService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="getServerInfo">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="getJobInfo">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="getAPIVersion">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="killJob">
<soap:operation soapAction=""/>
<input>
```

```
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="runJob">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
</binding>
<service name="RWWebService">
<port name="RWWebServicePort" binding="tns:RWWebServicePortBinding">
<soap:address location="http://localhost:8888/reports/rwwebservice"/>
</port>
</service>
</definitions>
```

19.3 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.

19.3.1 Using Oracle Enterprise Manager to Test RWWebService

To test the RWWebservice through Oracle Enterprise Manager, complete the following steps:

1. Log in to Oracle Enterprise Manager.
2. Navigate to your WebLogic Domain.
3. From the **WebLogic Domain** menu, select **Web Services** > **Test Web Service**

   The **Test Web Service page** is displayed.
4. Enter the URL of a WSDL and click **Parse WSDL**.
5. Select the webservice operation that you want to invoke.
6. Enter valid values in appropriate fields.
7. Click **Test Web Service**.

   The Webservice Response is displayed.
8. In case you want to edit the values, click the **Request** tab and complete the step 6 and 7.

The operations supported by the `RWWebService` endpoint are:

- getAPIVersion
- getServerInfo
- getJobInfo
- killJob

- runJob

19.3.1.1 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, for example, `runJob` return the response as an XML block embedded within the SOAP response.

To view the `getAPIVersion` response:

1. Navigate to the **Test Web Service** page in Oracle Enterprise Manager. See steps 1-3 in Section 19.3.1, "Using Oracle Enterprise Manager to Test RWWebService".
2. From the **Operation** drop-down list, select **getAPIVersion**.

   The Test page does not display any parameters.
3. Click **Test Web Service**.

   The SOAP response is displayed in the **Response** tab.

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">11.1.1.1.0</return>
    </ns1:getAPIVersionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

19.3.1.2 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. Navigate to the **Test Web Service** page in Oracle Enterprise Manager. See steps 1-3 in Section 19.3.1, "Using Oracle Enterprise Manager to Test RWWebService".
2. From the **Operation** drop-down list, select **getServerInfo.**

   The Test page displays the relevant parameter fields in the **Input Arguments** section.

3. Enter the Reports Server name (`arg0`) and `authId` (`arg1`).
4. Click **Test Web Service**.

   The SOAP response is displayed in the **Response** tab.

The following is a sample output of the `getServerInfo` operation:

```
<?xml version = '1.0' encoding = 'ISO-8859-1' standalone = 'yes'?>
<serverInfo name="repserv" version="11.1.1.1.0">
   <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="AverageResponseTime" value="2124"/>
      <property name="executionTimeToDate" value=""/>
   </performance>
</serverInfo>
```

19.3.1.3 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. Navigate to the **Test Web Service** page in Oracle Enterprise Manager. See steps 1-3 in Section 19.3.1, "Using Oracle Enterprise Manager to Test RWWebService".
2. From the **Operation** drop-down list, select **getJobInfo**.

   The Test page displays the relevant parameter fields in the **Input Arguments** section.
3. Enter the `jobId` (`arg0`), Reports Server name (`argm1`), and `authId` (`arg2`).
4. Click **Test Web Service**.

   The SOAP response is displayed in the **Response** tab.

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>
```

19.3.1.4 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. Navigate to the **Test Web Service** page in Oracle Enterprise Manager. See steps 1-3 in Section 19.3.1, "Using Oracle Enterprise Manager to Test RWWebService".
2. From the **Operation** drop-down list, select **killJob**.

   The Test page displays the relevant parameter fields.
3. Enter the `jobId` (`arg0`), Reports Server name (`arg1`), and `authId` (`arg2`).
4. Click **Test Web Service**.

   The SOAP response is displayed in the **Response** tab.

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>
```

19.3.1.5 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@oraDB
  ```

  > **Note:** The command line parameter cannot include `paramform=yes`. You have to pass the actual values for the parameter as part of the `commandLine` argument.

- *synchronous*: A Boolean object to indicate if the job should be run synchronously.

To view the `runJob` response:

1. Navigate to the **Test Web Service** page in Oracle Enterprise Manager. See steps 1-3 in Section 19.3.1, "Using Oracle Enterprise Manager to Test RWWebService".
2. From the **Operation** drop-down list, select **runJob**.

   The Test page displays the relevant parameter fields.
3. Enter the command line syntax (`arg0`), whether the job should run synchronously (T/F, Y/N) (`arg1`).
4. Click **Test Web Service**.

   The SOAP response is displayed in the **Response** tab.

The following is a sample output of a `runJob` operation:

```
<?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>
```

19.4 Using RWWebServiceUtil to Test RWWebService

Oracle Reports installation provides a sample `RWWebServiceUtil` webservice testing utility class. This is available in `ORACLE_HOME/reports/jlib/rwrun.jar`. RWWebServiceUtil can be used to test various operations supported by RWWebService.

The following procedure outlines the necessary steps involved in using this utility:

1. Include `rwrun.jar` in the classpath.
2. Run `RWWebServiceUtil` as a normal java program as following:

   ```
   $ORACLE_HOME\jdk\bin\java oracle.reports.rwclient.RWWebServiceUtil
   ```

3. It displays the following usage:

   ```
   -endpoint       url of the webservice
   -method         web service operation to invoke

   runJob,getJobInfo,getServerInfo,killJob,getAPIVersion

      Input Parameters for runJob
       -cmdline        command line to be used while submitting the job
       -sync           boolean value to specify if job should be submitted
                       in synchronous manner or asynchronous manner

      Input Parameters for getJobInfo
       -server         server to be used for processing the request
       -jobid          jobid in numeric format
       -authid         user/password for authentication, if server is secure

      Input Parameters for getServerInfo
       -server         server to be used for processing the request
       -authid         authid for authentication, if server is secure

      Input Parameters for killJob
       -server         server to be used for processing the request
       -jobid          jobid in numeric format
       -authid         authid for authentication, if server is secure
   ```

4. You can use this utility to submit jobs to reports server using the following command:

   ```
   $ORACLE_HOME\jdk\bin\java
   oracle.reports.rwclient.RWWebServiceUtil -endpoint
   http://yourmachine:port/reports/rwwebservice -method runjob -cmdline
   "report=test.rdf desformat=pdf destype=file desname=/tmp/output.pdf
   server=ReportsServer" -sync true
   ```

5. This submits the request to the server and returns a soap response in the following format:

   ```
   <?xml version = '1.0' encoding = 'UTF-8' standalone = 'yes'?>
      <serverQueues>
   ```

```
    <job id="2" queueType="past">
       <name>/home/vnanda/test.rdf</name>
       <type>report</type>
       <status code="4">Report  finished successfully.</status>
       <owner>RWUser</owner>
       <server>reportsserver</server>
       <destination>
          <desType>file</desType>
          <desName>/tmp/output.pdf</desName>
          <desFormat>pdf</desFormat>
          <file>output1.pdf</file>
       </destination>
       <timingInfo>
          <queued>Feb 17, 2009 9:41:36 AM</queued>
          <started>Feb 17, 2009 9:41:36 AM</started>
          <finished>Feb 17, 2009 9:41:38 AM</finished>
       </timingInfo>
    </job>
 </serverQueues>
```

6. Similarly, you can invoke other operations on `RWWebService` using the `RWWebServiceUtil`.

20

Creating Advanced Distributions

When you wish to define an advanced distribution for your report, you can design the distribution by developing a distribution XML filer. In this file, you can specify the destination and format of output for each section of a report. In one distribution XML file, you can specify many different destinations, including custom (pluggable) destinations that you design (see Section 20.3.9, "destype").

> **Note:** An example distribution XML file (`distribution.xml`) is shipped with Oracle Reports in the `/samples/demo` directory which can now be downloaded from OTN at `http://www.oracle.com/technetwork/middleware/reports/downloads/index.html`. You can reuse this file for your own purposes so that you do not have to create one from scratch.

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
- Using a Distribution XML File at Runtime
- Limitations with Using Distribution

20.1 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 e-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 to *burst* the report on each instance of the group to send each department's output to its related manager.

The distribution XML file simplifies 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.

In order to use the same report definition file to burst and distribute to data-driven formats such as XML and DELIMITEDDATA, as well as to layout-driven formats such as PDF and ENHANCEDSPREADSHEET, you must ensure the following requirements are met:

- The distribution XML file must specify the include element. For example:

  ```
  &lt;include src="mainSection"/&gt;
  ```

- The Repeat On property must be set appropriately for the section(s) specified in the distribution XML file.
- The section(s) specified in the distribution XML file in the report paper layout must not be empty.

20.2 Introduction to Distribution XML Files

This section discusses the use of XML files related to distribution:

- The distribution.dtd File
- Using Variables Within Attributes

20.2.1 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 on both Windows and UNIX:

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 you review the descriptions of the elements and attributes.

Note: Information provided in the distribution XML file is case-sensitive. You 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.

20.2.2 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 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 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"
```

20.3 Elements of a Distribution XML File

The elements of a distribution XML file include:

- destinations
- foreach
- mail
- body
- attach

- include
- file
- printer
- destype
- 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. Section 20.4, "Distribution XML File Examples" provides use cases that demonstrate the distribution XML file elements and attributes in typical scenarios.

20.3.1 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.

20.3.2 foreach

Example

```
<foreach>
  <mail id="a1" to="my_addressee@mycompany.com" subject="Fourth Quarter Results">
    <attach format="pdf" name="dept_&amp;&lt;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 tags 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 the following messages:

```
REP-177: Error while running in remote server
REP-34320: Report sections used in destination '<destination id>' do not repeat on
the same group
```

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>
```

20.3.3 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 through an outgoing SMTP-based mail server. Use it to specify the recipients, the subject, and the priority of the e-mail.

Between an open and close tag of the `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 20–1 lists and describes the attributes of the `mail` element.

Table 20–1 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.

Table 20–1 (Cont.) Attributes of the mail Element

Attribute	Valid Values	Description
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 20.2.2 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.
returnReceipt	true|false	**Default:** false Indication of whether the reply to individual or account should be notified when the e-mail is received.
organization	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 (*rwserver*.conf). This file has a pluginParam element where you can enter the name of a mail server. For example:

```
<pluginParam name="mailServer" value="%MAILSERVER%">
  <property name="enableSSL" value="yes"/>
</pluginParam>
```

See Chapter 7, "Configuring Oracle Reports Services".

20.3.4 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 tags of the `body` element 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>
…
```

The `body` element has three attributes: `srcType`, `format`, and `instance`, described in Table 20–2.

Table 20–2 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|delimiteddata	**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).

20.3.5 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>
```

or

```
<mail id="DEST1" to="tsmith@oracle.com" subject="Mail from dest" >
  <attach srcType="report" format="delimiteddata" name="myattach.txt"
    <include src="report"/>
  </attach>
</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

The attach element 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".

Table 20–3 lists and describes the attributes of the attach element.

Table 20–3 Attributes of the attach Sub-Element of mail

Attribute	Valid Values	Description
format	pdf|html|htmlcss|rtf|ascii|xml|delimiteddata|spreadsheet|enhancedspreadsheet|dflt	**Default:** pdf The format of the attached material, for example format="htmlcss". Required when srcType is report and the report format is other than pdf, the default; otherwise format is optional.
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 20.2.2 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>
```

20.3.6 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 `include`.

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 tag.

Table 20–4 describes the src attribute of the include element.

Table 20–4 Attributes of the include Sub-Element When Used with 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, you need not specify the report's name or location in the src attribute when src="report". *(path and) filename*: The preceding element must be either body or attach, with srcType=file. Provide the directory path and file name or just a file name if the file is located in a directory listed in the REPORTS_PATH environment variable. *Other values*: When the preceding body or attach element specifies srcType=report, specify the entire report (report) or provide the section(s) of the report to be included in the body or to be attached (for example, headerSection,mainSection, or trailerSection).

20.3.7 file

Example

On Windows

```
<file id="a7" name="c:\management\reports\report.pdf" format="pdf">
  <include src="report"/>
</file>
```

or

```
<file id="MyFiles2" name="c:\My_Departments_Report.txt" format="delimiteddata">
  <include src="report"/>
</file>
```

On UNIX

```
<file id="a7" name="/management/reports/report.pdf" format="pdf">
  <include src="report"/>
```

```
</file>
```

or

```
<file id="MyFiles2" name="/My_Departments_Report.txt" format="delimiteddata">
  <include src="report"/>
</file>
```

On Windows or UNIX

```
<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. The `file` element has one sub-element: include. There must be at least one `include` sub-element and there may be more between an open and close tag of the `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 src="report"/>
  </file>
</foreach>
```

In this example, all report sections (header, main, and trailer) must repeat on the same group instance (for example, `department_id`).

Table 20–5 lists and describes the attributes of the `file` element.

Table 20–5 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.

Table 20–5 (Cont.) Attributes of the file Element

Attribute	Valid Values	Description
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\q4sales.pdf" **UNIX:** name="reports/q4sales.pdf Can also contain variable values that reference columns used in the associated report. See Section 20.2.2 for more information.
format	pdf|html|htmlcss|rtf|ascii|xml|delimiteddata|spreadsheet|enhancedspreadsheet|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).

20.3.8 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. The printer element has one sub-element: include. There must be at least one include sub-element and there may be more between the open and close tags of the 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 src="report"/>
  </printer>
</foreach>
```

In this example, all report sections (header, main, and trailer) must repeat on the same group instance (for example, `department_id`).

Table 20–6 lists and describes the attributes of the `printer` element.

Table 20–6 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.
`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 20.2.2 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`).

20.3.9 destype

Example

```
<destype id="acustom1" name="fax">
  <include src="headerSection"/>
  <property name="number" value="914925551212"/>
</destype>
```

See Section 20.4, "Distribution XML File Examples" for examples of using the `destype` element in a distribution XML file to specify distribution to the following destinations: Oracle Portal, FTP, WebDAV, and fax.

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 a fax machine or an FTP site. You also use destype to specify distribution to a portal created with Oracle Portal. The destype element allows for the use of two sub-elements: property and include. At least one include is required.

> **Note:** The inclusion of a custom destination type requires that you have a defined distribution handler in place to push report content to the custom output destination. Build a custom destination type through the Oracle Reports Services Destinations API.
>
> For more information about the available APIs for Oracle Reports, refer to Oracle Reports Java API Reference on the Oracle Technology Network (OTN) at (http://www.oracle.com/technetwork/middleware/reports/overview/index.html).

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 (for example, a separate fax). For example, if you group a report on department_id, and there are four departments, you can 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 src="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 (for example, fax_number).

Custom destination types also have a set of related attributes. These are expressed within the destype tag. For example, the id, name, and instance attributes are expressed:

```
<foreach>
  <destype id="a1" name="name_of_destination_type" instance="all">
    <include src='report'/>
  </destype>
</foreach>
```

Table 20–7 lists and describes the attributes of the destype element.

Table 20–7 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.

Table 20–7 (Cont.) Attributes of the destype Element

Attribute	Valid Values	Description
name	string	Required. The name of the custom destination. For example, for a fax, this might be: name="fax" For a portal built with Oracle 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.

20.3.10 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 Oracle Reports Services. How you specify properties is entirely dependent on the requirements of your custom destination.

20.4 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
- destype Examples

20.4.1 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

20.4.1.1 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.

20.4.1.2 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>
```

20.4.1.3 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>
```

20.4.1.4 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 20.2.2), 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 (for example, department 40).

Oracle Reports supports PDF encryption in distribution and bursting of Reports. With this feature, you can secure your Reports output through PDF Security in the following way.

```
<foreach>
  <file id="a10" name="department_&amp;&lt;department_id&gt;.pdf" instance="this">
    <property name="pdfuser" value="&amp;&lt;department_id&gt;"/>
     <include src="mainSection"/>
  </file>
</foreach>
```

To open the generated PDF output, you must provide the password as `departmentid`.

20.4.1.5 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.

20.4.1.5.1 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>
```

20.4.1.5.2 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>
```

20.4.2 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

20.4.2.1 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>
```

20.4.2.2 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.

20.4.2.3 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>
```

20.4.2.4 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.

20.4.2.4.1 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>
```

20.4.2.4.2 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>
```

20.4.2.5 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>
```

20.4.2.6 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>
```

Oracle Reports 12*c* Release (12.2.1.3) supports PDF encryption in distribution and bursting of reports. With this feature, you can secure Reports output through PDF Security in the following way:

```
<foreach>
 <mail id="mgr1090" to="&amp;&lt;manager&gt;@mycompany.com">
   <property name="pdfuser" value="&amp;&lt;manager&gt;" />
  <attach format="pdf" name="attach.pdf" srcType="report" instance="this">
   <include src="mainSection"/>
  </attach>
 </mail>
</foreach
```

20.4.3 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, and so on. You will end up with only one report, that of the final department. 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

20.4.3.1 File for Whole Report

This example will yield one file named `report.pdf` that contains the entire report.

20.4.3.1.1 Windows

```
<file id="a1" name="c:\reports\report.pdf" format="pdf">
  <include src="report"/>
</file>
```

20.4.3.1.2 UNIX

```
<file id="a1" name="/reports/report.pdf" format="pdf">
  <include src="report"/>
</file>
```

20.4.3.2 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 src="headerSection"/>
  <include scr="mainSection"/>
</file>
```

20.4.3.3 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>
```

20.4.3.4 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>
```

20.4.4 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.

20.4.4.1 Print Whole Report

In this example, the entire report will be sent to the specified printer.

20.4.4.1.1 Windows

```
<printer id="a80" name="\\neptune\prtr20">
  <include src="report"/>
</printer>
```

20.4.4.1.2 UNIX

```
<printer id="a80" name="10th_floor_printer">
  <include src="report"/>
</printer>
```

20.4.4.2 Print Two Sections of a Report

In this example, two sections of a report will be sent to the printer.

20.4.4.2.1 Windows

```
<printer id="a1" name="\\neptune\prtr20">
  <include src="headerSection"/>
  <include src="mainSection"/>
</printer>
```

20.4.4.2.2 UNIX

```
<printer id="a1" name="10th_floor_printer">
  <include src="headerSection"/>
  <include src="mainSection"/>
</printer>
```

20.4.4.3 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.

20.4.4.3.1 Windows

```
<foreach>
 <printer id="prt20" name="\\neptune\prtr20" instance="all">
  <include src="report"/>
 </printer>
</foreach>
```

20.4.4.3.2 UNIX

```
<foreach>
<printer id="prt20" name="10th_floor_printer" instance="all">
<include src="report"/>
</printer>
</foreach>
```

20.4.4.4 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`.

20.4.4.4.1 Windows

```
<foreach>
 <printer id="prt20" name="\\neptune\prtr20" instance="this">
  <include src="mainSection"/>
  <include src="trailerSection"/>
 </printer>
</foreach>
```

20.4.4.4.2 UNIX

```
<foreach>
 <printer id="prt20" name="10th_floor_printer" instance="this">
   <include src="mainSection"/>
   <include src="trailerSection"/>
 </printer>
</foreach>
```

20.4.4.5 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 (for example, 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_name` is tied to a department, 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

20.4.5 destype Examples

You can use `destype` to define a custom destination or pluggable destination that can be used by Oracle Reports during distribution. See Section 20.3.9, "destype". The examples in this section include the following destinations:

- Oracle Portal Destination
- FTP Destination
- WebDAV Destination
- Fax Destination

20.4.5.1 Oracle Portal Destination

This example shows the generic tag structure for sending report output to the Oracle Portal destination. When you push report output to Oracle Portal using `DESTYPE=ORACLEPORTAL`, the report output is created in the PAGEGROUP folder.

> **See Also:** Appendix A, "Command-Line Keywords" for more information on the properties shown in the examples.

```
<destinations>
  <destype id="customforPortal" name="oraclePortal">
    <property name="outputpage" value="sample_report"/>
    <property name="statuspage" value="Reports_Status"/>
    <property name="pagegroup" value="REPORTS_OUTPUT"/>
    <property name="itemtitle" value="MyReport"/>
    <include src="report"/>
  </destype>
</destinations>
```

20.4.5.2 FTP Destination

This example shows the generic tag structure for sending report output to the FTP destination.

```
<destinations>
 <foreach>
  <destype id="ftp1" name="ftp" instance="this" format="pdf">
   <property name="desname"
    value="ftp://username:passwd@ftpServer/dir/myreport_&amp;&lt;
    DEPARTMENT_NAME&gt;.pdf"/>
    <include src="mainSection"/>
  </destype>
 </foreach>
</destinations>
```

20.4.5.3 WebDAV Destination

This example shows the generic tag structure for sending report output to the WebDAV destination.

```
<destinations>
 <foreach>
  <destype id="webdav1" name="webdav" instance="this" format="pdf">
   <property name="desname"
    value="http://user:passwd@WebDAVServer/dir/myreport_&amp;&lt;
    DEPARTMENT_NAME&gt;.pdf"/>
    <include src="mainSection"/>
  </destype>
 </foreach>
</destinations>
```

20.4.5.4 Fax Destination

This example shows the generic tag structure for sending report output to the fax destination.

```
<destype id="faxdest" name="fax">
  <property name="number" value="123456789"/>
  <include src="report"/>
</destype>
```

Alternatively, for ease of use, you can specify a custom, more specific tag structure:

```
<fax id="faxdest" number="123456789">
  <include src="report"/>
</fax>
```

Note: All you must do after you modify the `distribution.xsl` file is, save it back to the same location under the same file name. Oracle Reports will automatically look for this file when resolving distributions.

20.5 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. Include the options:

`destination=`*`filename`*`.xml distribute=yes`

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:

```
destination=c:\%ORACLE_HOME%\reports\distribution\filename.xml distribute=yes
```

For UNIX, you might specify:

```
destination=$ORACLE_HOME/reports/distribution/filename.xml distribute=yes
```

For example, the full command in a URL would be similar to:

```
http://your_server:port/reports/rwservlet?report=rep.jsp&userid=db_credentials
&destination=$ORACLE_HOME/reports/distribution/distribution.xml&distribute=yes
```

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 Oracle 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 preceding example, 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 18, "Running Report Requests".

20.6 Limitations with Using Distribution

This section outlines the limitations with using distribution in Oracle Reports:

- Delimited Output
- Dynamic Format Values

20.6.1 Delimited Output

- Report bursting and distribution does not support Delimited output format. You cannot specify `DELIMITED` as an output format in a distribution XML file or in the Distribution dialog box.

> **Note:** You can distribute a report in DelimitedData output format, specified either in a distribution XML file or in the Distribution dialog box in Reports Builder

20.6.2 Dynamic Format Values

XML distribution supports only static values for the `format` attribute (as seen in `distribution.dtd`). Thus, you cannot specify lexical parameters (to be resolved at runtime) for the format attribute. Hence the format cannot be dynamically determined either for the entire report or for a specific section.

21

Using Event-Driven Publishing

Modern business processes often require the blending of automation into the work environment through 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 Reports Services includes a scheduling mechanism that enables the invocation of reports on a scheduled basis without requiring additional user interaction. But this leaves 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 Oracle 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 Oracle Reports Services from within the database.

This chapter provides an overview of the Event-Driven Publishing API and includes examples of its use. It includes the following sections:

21.1 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.

21.1.1 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 **SRW_ParamList** 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 **SRW_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 Oracle Reports Services Security and Oracle 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 Event-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.

21.1.2 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

21.1.2.1 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 fourth—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`.

21.1.2.2 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;
```

21.1.2.3 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.

21.1.3 Including non-ASCII Characters in Parameter Names and Values

To use non-ASCII characters in user parameter names and values when using the Event-Driven Publishing API, you must include in your parameter list a parameter called DEFAULTCHARSET, with its value set to a valid character set name. This character set name can be specified with either the database's NLS_CHARACTERSET (for example, JA16SJIS) or IANA-defined character set name (for example, WINDOWS-31J). You must also ensure that the value of the DEFAULTCHARSET parameter matches the defaultcharset parameter specified in the rwservlet.properties file. Oracle Reports Services encodes non-ASCII user parameter names and values using the character set specified by DEFAULTCHARSET, allowing you to use the Event-Driven Publishing API for reports with non-ASCII characters in parameter names and values.

Note: If you do not add a parameter called DEFAULTCHARSET to your parameter list, Oracle Reports Services encodes your user parameter names and values using the database's NLS_CHARACTERSET.

21.1.4 Submitting 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 Oracle Reports Servlet (rwservlet). 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 Oracle Reports Servlet (rwservlet) you will use to process the request.
- SERVER identifies the Reports Server to be used in conjunction with Oracle Reports Servlet (rwservlet).
- 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 (through 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.

21.1.5 Checking 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 executing the rwservlet Web command showjobs.

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.

21.1.6 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.

21.2 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 through `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 Oracle Reports Services installation.

21.3 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 enables 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
DECLARE
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 key map file (see Section 18.14, "Using a Key Map 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 through 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 Oracle Reports Services installation.

21.4 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 Oracle Reports Services does not fit your needs.

21.4.1 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 Oracle 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.

21.4.2 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 through 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 Oracle Reports Services installation.

21.4.3 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 timeout defined in the second line of code. This timeout defines the amount of seconds the dequeue procedure should wait for a message before creating a timeout 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.

22

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` keyword 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 Oracle Reports Builder.

Oracle 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 through Oracle Reports XML customization means that any data model that can be created with Oracle 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:

- 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.
- For descriptions of selected Oracle Reports XML tags, see topic "Oracle Reports XML tags" in the **Reference** section of the *Oracle Reports online Help*.
- 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.

Note: For reports developed in a release prior to Oracle Reports 10*g* Release 2 (10.1.2) patch 2, you may find the PL/SQL package specification or body is missing when opening the XML reports. In this case, either:

If the RDF exists, regenerate the XML output file using Oracle Reports 10*g* Release 2 (10.1.2) patch 2 or later.

Or, edit the XML to add `type="packageSpec"` and/or `type="packageBody"` in the `function` element, as follows:

```
<programUnits>
  <function name="a" type="packageSpec">
    <textSource>
      <![CDATA[PACKAGE a IS
          function lire return date ;
        END a;]]>
    </textSource>
  </function>
  <function name="a" type="packageBody">
    <textSource>
      <![CDATA[PACKAGE BODY a IS
          function lire return date is
            c2 date;
          ...
        END;]]>
    </textSource>
  </function>
  <function name="cf_1formula" returnType="date">
    ...
  </function>
</programUnits>
```

22.1 Customization Overview

By using the Oracle Reports XML tags, you can customize reports created using Oracle 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 Oracle 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.

2. Store the XML customization in a location that is accessible to Oracle 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: Section A.5.21, "CUSTOMIZE"

Note: For a description of the `SRW` built-in package, including the `SRW.APPLY_DEFINITION` built-in procedure, see the *Oracle Reports online Help*.

22.2 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

22.2.1 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`.

Note: To apply an XML customization file to modify an existing report trigger or to create a new report trigger, you must specify the relevant trigger attribute of the `<report>` tag:

For example, to modify or create a Before Report trigger, use the `beforeReportTrigger` attribute:

```
<report DTDVersion="9.0.2.0.0" beforeReportTrigger="BeforeReport">
```

If you do not specify this attribute when you want to apply an XML customization file to modify or create a report trigger, the report trigger PL/SQL code will be treated as a local (independent) function when the XML customization file is applied to your report.

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 need not specify them. Examples of the `data` and `layout` elements are provided in the following sections.

22.2.2 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.

22.2.3 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>`.

22.2.4 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 22–1:

Table 22–1 Values for the operator Attribute

Value	*Usage*
`eq`	equal
`lt`	less than
`lteq`	less than or equal to
`neq`	not equal to
`gt`	greater than
`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 (`/>`).

22.2.5 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.

22.2.6 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 `<field></field>` tag pair. If there is both a global and local `labelAttribute` element (one outside and one inside the `<field></field>` tag pair), the local overrides the global.

22.2.7 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`.

22.3 Creating XML Data Models

Oracle 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, Oracle 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.

22.3.1 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>
```

22.3.2 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.

22.3.3 Deleting a Data Link object from Data Model

There may be a situation when you may need to delete or clear one or more data link objects from the data model. To do so:

1. Go to the desired data model.
2. Select the data link like object that you want to clear and right-click using the mouse.
3. Select the Delete/Clear option from the menu that appears on mouse click.

22.3.4 Creating Group Hierarchies Within Each Data Source

With Oracle 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>
```

22.3.5 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>
```

22.3.6 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>
```

22.3.7 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>
```

22.4 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 Section 22.4.1, "Applying an XML Report Definition at Runtime" for more information.

  Note: Oracle Reports does not support XML customizations of REP files.

- You can run an XML report definition by itself (without another report) by specifying the `REPORT` (or `MODULE`) command line keyword. Refer to Section 22.4.2, "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 Section 22.4.3, "Performing Batch Modifications" for more information.

The following sections describe each of the cases in more detail and provide examples.

22.4.1 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 Section 22.4.3, "Performing Batch Modifications" for more information about using `CUSTOMIZE` with `rwconverter`.

22.4.1.1 Applying One XML Report Definition

The following command line sends a job request to Oracle Reports Services and applies an XML report definition, `emp.xml`, to an RDF file, `emp.rdf`. In this example, the `CUSTOMIZE` keyword refers to a file located in a Windows directory path. For UNIX, specify the path according to UNIX standards (that is, `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
```

22.4.1.2 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 Oracle 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 value demonstrates a directory path to files stored on a Windows platform. For UNIX, use that platform's standard for specifying directory paths (that is, 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
```

22.4.1.3 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-in procedures in the Before Parameter Form or After Parameter Form trigger. The following sections provide examples of these built-in procedures.

> **Note:** For a description of the SRW built-in package, including the SRW.APPLY_DEFINITION and SRW.ADD_DEFINITION built-in procedures, and more information about report triggers, see the *Oracle Reports online Help*.

22.4.1.3.1 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 Before Parameter Form or After Parameter Form 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.

22.4.1.3.2 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 After Parameter Form trigger of a report called videosales_custom.rdf.

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

- Conditionally highlights fields based upon parameter values entered by the user at runtime.
- Changes number format masks based upon parameter values entered by the user at runtime.

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="LNNNNNNNNNNN0"/>');
```

```
    SRW.ADD_DEFINITION('<field name="F_SumTOTAL_COSTPerCITY"
      source="SumTOTAL_COSTPerCITY" formatMask="LNNNNNNNNNN0"/>');
    SRW.ADD_DEFINITION('</section>');
    SRW.ADD_DEFINITION('</layout>');
  END IF;
  SRW.ADD_DEFINITION('</report>');
  SRW.APPLY_DEFINITION;
  SRW.ADD_DEFINITION('<report name="vidsales_hilite_costs" author="Generated"
    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;
```

22.4.2 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 Oracle 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 command line keyword.

22.4.3 Performing Batch Modifications

If you have a large number of reports that has to be updated, 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 Oracle Reports Builder, you can run rwconverter once and make the same change to a large number of reports at once.

Note: The rwconverter.bat file uses the start keyword to start the rwconverter process. If the start keyword is specified, then the process starts in asynchronous mode. The spawned rwconverter continues execution until completion. For using batch conversion of a large number of rdfs, all the processes start simultaneously. Therefore, it is recommended that you modify the rwconverter.bat file manually to remove the start keyword for running rwconverter in batch mode.

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 value demonstrates a directory path to files stored on a Windows platform. For UNIX, use that platform's standard for specifying directory paths (that is, forward slashes instead of backward).

22.5 Debugging XML Report Definitions

The following features are available to help you debug your XML report files:

- XML Parser Error Messages
- rwbuilder
- Writing XML to a File for Debugging

22.5.1 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://www.oracle.com/technetwork/database/features/xmldb/overview/xdkhome-092156.html). Search for *XML parser* or *XDK*. Information is also available in the documentation that came with your Oracle Database.

22.5.2 rwbuilder

When designing an XML report definition, it is sometimes useful to open it in Oracle Reports Builder. In Oracle 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 Oracle 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 Oracle 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 Oracle 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` option specifies a directory path to files stored on a Windows platform. For UNIX, use that platform's standard for specifying directory paths (that is, forward slashes instead of backward slashes).

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

22.5.3 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 must declare a variable of type `TEXT_IO.File_Type`. For example:

```
custom_summary  Text_IO.File_Type;
```

You must also 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);
```

Part V

Globalization Support and Bidirectional Support

Part V provides information about Reports-related globalization support settings and bidirectional support:

23

Implementing Globalization 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. Oracle Reports Services includes the support that you require to address any issues related to these considerations: Globalization support for character sets and bidirectional support for text reading order.

Globalization support makes it possible to design applications that can be deployed in several different languages. Oracle supports most European, Middle Eastern, and Asian languages. globalization support 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 globalization support architecture, including globalization support settings relevant to Reports; explains how to specify character sets in a JSP; and offers information on bidirectional, Unicode, and translation support. It includes the following main sections:

23.1 Globalization Support Architecture

Globalization support architecture consists of two parts:

23.1.1 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.

23.1.2 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 globalization support 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.

23.2 Globalization Support Environment Variables

Globalization support environment variables are automatically set to default values during Oracle Fusion Middleware installation.

> **Note:** With the environment switching feature, you are not limited to the default environment set at the time of installation, and you can configure multiple environments, including language settings, for a single Reports Server. For more information, refer to Section 7.2.2, "Dynamic Environment Switching".

Table 23–1 lists and describes globalization support-related environment variables that are relevant to Oracle Reports Services.

Table 23–1 Oracle Reports ServicesEnvironment Variables for Globalization Support

Variable	Description
NLS_LANG	Relevant to Oracle Reports Services. The language settings used by Oracle Reports Services. See Section 23.2.1, "NLS_LANG Environment Variable".
DEVELOPER_NLS_LANG	The language for Oracle Reports Builder. If not set, then NLS_LANG default values are used. See Section 23.2.2, "DEVELOPER_NLS_LANG and USER_NLS_LANG Environment Variables".
USER_NLS_LANG	The language for Reports Runtime. If not set, then NLS_LANG default values are used. See Section 23.2.2, "DEVELOPER_NLS_LANG and USER_NLS_LANG Environment Variables".

23.2.1 NLS_LANG Environment Variable

The NLS_LANG environment variable specifies the language, territory, and character set settings to be used by Oracle 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 Fusion Middleware. Refer to Section 23.2.1.1, "Defining the NLS_LANG Environment Variable" for more information about changing the environment variable after installing Oracle Fusion Middleware.

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. See Section 23.2.1.2, "Defining the Language and Territory".

- `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. See Section 23.2.1.2, "Defining the Language and Territory".

- `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. See Section 23.2.1.3, "Defining the Character Set".

> **Note:** When you use features like Oracle Portal Security, Portal Destination, and Job Status Repository, the JDBC database connections made by Oracle Reports Services may override the initial NLS_LANG setting. This change may in turn affect the behavior of the running report, such as aliasing PDF output in Asian languages. On UNIX platforms, you can work around this issue by using the environment switching functionality to dynamically set the environment for reports, as described in Section 7.2.2, "Dynamic Environment Switching".

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 `rwservlet` pages such as `showjobs`, `showenv`, online Help, and error messages are delivered from the middle tier machine's locale (or `LANG` on UNIX) and not `NLS_LANG`. For example, if 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.

23.2.1.1 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.

23.2.1.1.1 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. Expand the **Oracle** node, then click your Oracle Reports Services **HOME** node 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**.

23.2.1.1.2 UNIX To define the `NLS_LANG` environment variable on the UNIX platform, set it in the shell script `reports.sh`, located in your `ORACLE_INSTANCE/config/reports/bin` directory.

23.2.1.2 Defining the 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 Oracle Reports Services messages and message from the Oracle Database.
- 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).

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
REP-0110: Unable to open file.

French:
ORA-00942: table ou vue inexistante
REP-0110: Impossible d'ouvrir le fichier
```

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.

23.2.1.3 Defining the Character Set

The character set component of the globalization support 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.

23.2.1.3.1 Character Set Design Considerations If you are designing a multilingual application, or even a single-language application that runs with multiple character sets, you must determine the character set most widely used at runtime and then set the `NLS_LANG` environment variable to that particular character set.

If you design 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 design-time character set, then question marks appear in place of the unrecognized characters.

23.2.1.3.2 Font Aliasing Considerations There may be situations where you create a multilingual 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 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, Oracle 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:

```
ARIAL.....=ARIAL.....WE8ISO8859P1
```

This example specifies that any Arial font should be mapped to the same value, but with the WE8ISO8859P1 character set.

For more information about font aliasing and uifont.ali, see Section 11.1.2.1, "Font Aliasing".

23.2.2 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 Fusion Middleware 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
```

If these environment variables are not specifically set, then NLS_LANG default values will be used. Use the DEVELOPER_NLS_LANG and USER_NLS_LANG environment variables in lieu of the NLS_LANG environment variable in the following situations:

- You prefer to use Reports Builder in a particular language (for example, 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 Oracle Reports Builder and Reports Runtime.
- You are creating an application to run in a language for which a local language version of Oracle Reports Builder is not currently available.

23.3 Specifying a Character Set in a JSP or XML File

In Oracle Reports, Web-report templates are configured by default for Western European character encoding. For other languages, you must specify the character encoding for a JSP file by using both the charset attribute of the <meta> tag and the <%@page%> page directive.

To dynamically associate the appropriate character encoding with the JSP file, you can make the following modifications:

1. In the directory *oracle_home*/reports/templates/, edit the files rw*.html and blank_template.jsp:
   a. Modify the page directive to read

```
<%@ page contentType="text/html;charset=yourIANAencoding" %>
```

where,

yourIANAencoding is the IANA character encoding that corresponds to the character encoding part of your NLS_LANG environment variable.

b. Modify the <meta> tag inside the <head> tag to read:

```
<meta http-equiv="Content-Type"
content="text/html;charset=yourIANAencoding" />
```

2. In the directory *oracle_home*/reports/templates/, edit the file template.xsl:

a. Modify the <xsl:output> tag to read:

```
<xsl:output
     method="jsp"
     indent="yes"
     encoding="yourIANAencoding"
   />
```

where

yourIANAencoding is the IANA encoding that corresponds to the character encoding part of your NLS_LANG environment variable.

b. Add the following page directive:

```
<%@ page contentType="text/html;charset=yourIANAencoding" %>
```

c. Add or modify the <meta> tag inside the <head> tag:

```
<meta http-equiv="Content-Type"
content="text/html;charset=yourIANAencoding" />
```

where

yourIANAencoding is the IANA encoding that corresponds to the character encoding part of your NLS_LANG environment variable.

The following example specifies a Japanese character set:

```
<%@ page contentType="text/html;charset=Shift_JIS" %>
<META http-equiv="Content-Type" content="text/html;charset=Shift_JIS">
```

Note: To set the character set in a paper layout report that you plan to use to generate XML, you must include a character set for the report's XML Prolog Value property:

```
<?xml version="1.0" encoding="&Encoding" ?>
```

&Encoding is then replaced at runtime with the appropriate setting.

The values expressed for the character set should call a character set that is compatible with the one specified for Oracle Reports Services. The values for character sets used on the Web (IANA-defined character sets) are different from the values expressed in the NLS_LANG environment variable. Table 23–2 lists commonly used IANA-defined character sets for the charset parameter:

Note: IANA-defined character set values are not case-sensitive. You can enter them in uppercase or lowercase.

Table 23–2 Valid Values for the IANA-Defined Character Sets

Languages	Valid IANA-Defined Character Sets
AMERICAN	ISO-8859-1, ISO-8859-15, windows-1252, US-ASCII, UTF-8
ARABIC	ISO-8859-6, windows-1256, UTF-8
ASSAMESE	UTF-8
BANGLA	UTF-8
BENGALI	UTF-8
BRAZILIAN PORTUGESE	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
BULGARIAN	ISO-8859-5, windows-1251, KOI8-R, UTF8
CANADIAN FRENCH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
CATALAN	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
CROATIAN	ISO-8859-2, windows-1250, UTF-8
CZECH	ISO-8859-2, windows-1250, UTF-8
DANISH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
DUTCH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
EGYPTIAN	ISO-8859-6, windows-1256, UTF-8
ENGLISH	ISO-8859-1, ISO-8859-15, windows-1252, US-ASCII, UTF-8
ESTONIAN	ISO-8859-4, ISO-8859-13, windows-1257, UTF-8
FINNISH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
FRENCH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
GERMAN DIN	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
GERMAN	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
GREEK	ISO-8859-7, windows-1253, UTF-8
GUJARATI	UTF-8
HEBREW	ISO-8859-8-I, windows-1255, UTF-8
HINDI	UTF-8
HUNGARIAN	ISO-8859-2, windows-1250, UTF8
ICELANDIC	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
INDONESIAN	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
ITALIAN	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
JAPANESE	EUC-JP, Shift_JIS, UTF-8
KANNADA	UTF-8
KOREAN	EUC-KR, UTF-8
LATIN AMERICAN SPANISH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
LATVIAN	ISO-8859-4, ISO-8859-13, windows-1257, UTF-8
LITHUANIAN	ISO-8859-4, ISO-8859-13, windows-1257, UTF-8
MALAY	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
MALAYALAM	UTF-8

Table 23–2 (Cont.) Valid Values for the IANA-Defined Character Sets

Languages	Valid IANA-Defined Character Sets
MARATHI	UTF-8
MEXICAN SPANISH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
NORWEGIAN	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
ORIYA	UTF-8
POLISH	ISO-8859-2, windows-1250, UTF8
PORTUGESE	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
PUNJABI	UTF-8
ROMANIAN	ISO-8859-2, windows-1250, UTF-8
RUSSIAN	ISO-8859-5, windows-1251, KOI8-R, UTF-8
SIMPLIFIED CHINESE	GBK, GB18030, UTF-8
SLOVAK	ISO-8859-2, windows-1250, UTF-8
SLOVENIAN	ISO-8859-2, windows-1250, UTF-8
SPANISH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
SWEDISH	ISO-8859-1, ISO-8859-15, windows-1252, UTF-8
TAMIL	UTF-8
TELUGU	UTF-8
THAI	TIS-620, UTF-8
TRADITIONAL CHINESE	Big5, Big5-HKSCS, UTF-8
TURKISH	ISO-8859-9, windows-1254, UTF-8
UKRANIAN	ISO-8859-5, windows-1251, KOI8-U, UTF-8
VIETNAMESE	windows-1258, UTF-8

23.4 Bidirectional Support

Bidirectional support enables you to design applications in those languages whose natural writing direction is right to left, such as Middle Eastern and North African languages. 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 (for example, right to left or left to right) .
- Alignment, which includes switching point-of-origin from upper left to upper right.

When you are designing bidirectional applications, you might want to use the globalization support environment variables `DEVELOPER_NLS_LANG` and `USER_NLS_LANG` rather than inheriting the `NLS_LANG` settings. 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. For more information, refer to Section 23.2, "Globalization Support Environment Variables".

On UNIX, you may continue to see misalignment of right-aligned text in PDF output for languages that read right to left. To work around this issue, use fixed width fonts instead of variable width fonts. For example, Miriam Fixed True Type font (Hebrew) is a fixed width font available on Windows 2000, and can be used for font subsetting on UNIX platforms to correct any font alignment issues with Hebrew fonts. For more information about resolving font issues across different platforms, see Chapter 12, "Font Model and Cross-Platform Deployment".

23.4.1 Enhanced BIDI Reshaping

Oracle Reports provides an Enhanced BIDI reshaping mechanism using the XDO APIs. This feature uses a new environment variable, REPORTS_ENHANCED_BIDIHANDLING, which specifies whether or not to use the new BIDI reshaping mechanism. You must set the value of this environment variable to YES if you want to use the new enhanced BIDI algorithm. If you do not set the environment variable, or if you set the value of the environment variable to NO, it uses the old mechanism.

If you set the value of REPORTS_ENHANCED_BIDIHANDLING to YES:

- All the values that were supported in the old BIDI reshaping mechanism are supported for REPORTS_ARABIC_NUMERAL environment variable.
- The following values are supported for REPORTS_BIDI_ALGORITHM environment variable:
  - UNICODE
  - ORACLE
  - UNICODE_VARIANT
- The default value is UNICODE.

For more information on how to use this environment variable, see Section B.1.47, "REPORTS_ENHANCED_BIDIHANDLING".

23.5 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 through the Internet.

This section discusses the following aspects of Unicode in Oracle Reports:

- Unicode Support
- Unicode Font Support
- Enabling Unicode Support

23.5.1 Unicode Support

Oracle Reports Services provides Unicode support. On UNIX platforms, Unicode support has certain limitations; for example:

- Unicode is not supported in PostScript output format on UNIX.
- In other bitmap output formats, such as PDF and RTF, you may observe font issues such as character misalignment on UNIX.

For information on how to resolve such issues, refer to Section 12.2.2, "Fixing Font-Related Issues".

If you use Unicode, you are able to display multiple languages, both single-byte languages such as Western European, Eastern European, 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=JAPANESE_JAPAN.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 character set portion of the `NLS_LANG` environment variable must be set to `UTF8`, along with the appropriate *`language_territory`* setting. For example:

```
NLS_LANG=JAPANESE_JAPAN.UTF8
NLS_LANG=GERMAN_GERMANY.UTF8
NLS_LANG=AMERICAN_AMERICA.UTF8
```

Unicode capability gives the application developer and end user the ability to display multilingual text in a report. This includes text from a database containing Unicode characters, multilingual boilerplate text, text in graphical user interface (GUI) objects, text input from the keyboard, and text from the clipboard.

> **Note:** If you develop applications for the Web, then you can use Unicode because of the Unicode support provided by Java through the browser.

23.5.2 Unicode Font Support

To enter text in a particular language, you must be running a version of the operating system that supports that language. Also, depending on the output format type, Oracle Reports Services relies on the operating system for the font for different languages, as described in Chapter 9, "Managing Fonts in Oracle Reports".

Windows provides True Type Big Fonts. These fonts contain the characters necessary to display or print text from more than one language. For example, if you try to type, display, or print Western European, Central European, and Arabic text on a field and see unexpected characters, then you are probably not using a Big Font. Big Fonts for single-byte languages provided by Microsoft Windows are Arial, Courier New, and Times New Roman. See http://www.microsoft.com/typography/fonts/default.aspx.

Oracle provides two Unicode fonts for Western European, Central European, Cyrillic, Greek, Turkish, Hebrew, Arabic, Baltic, Vietnamese, Thai, Simplified Chinese, Japanese, Korean, and Traditional Chinese:

- Albany WT fonts (proportional width) are available in Oracle Fusion Middleware 12c (12.2.1.3.0) MRUA CD.
- Andale Duospace WT fonts (fixed width) can be downloaded from My Oracle Support (http://support.oracle.com). The patch ID is 2638552.

Third-party Unicode fonts are also available.

23.5.3 Enabling Unicode Support

To enable Unicode support, set the NLS_LANG environment variable as follows:

```
NLS_LANG=language_territory.UTF8
```

Refer to Section 23.2, "Globalization Support Environment Variables" for more information about environment variables.

23.6 Translating Applications

In any Oracle Reports Services application, you see many types of messages, including:

- Error messages from the database
- Runtime error messages produced by Oracle Reports Services
- Messages and boilerplate text defined as part of the application

If the NLS_LANG 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 your Oracle Developer Suite documentation CD and on the Oracle Technology Network (http://www.oracle.com/technetwork/developer-tools/forms/overview/index.html).

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 SRW.MESSAGE built-in procedure,

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(index_no NUMBER)
RETURN CHAR
IS
   msg CHAR(80);
BEGIN
   IF index_no = 1001 THEN
      msg := 'L''impiegato che Voi cercate non esiste...';
   ELSIF index_no = 1002 THEN
      msg := 'Lo stipendio non puo essere minore di zero.';
   ELSIF  ...
   .
   .
   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:

```
SRW.MESSAGE(1001,nls_appl_mesg(1001));
```

> **Note:** For a description of the `SRW` built-in package, including the `SRW.MESSAGE` built-in procedure, see the *Oracle Reports online Help*.

To change the application to another language, simply replace the PL/SQL library containing the `nls_appl_mes` function with a library of the same name containing the `nls_appl_mesg` function with translated text.

23.7 Troubleshooting Globalization Issues

To help resolve globalization issues that may occur in your multilingual applications, this section provides the following troubleshooting information:

- Embedding a Character Set in a JSP file Dynamically
- Setting Globalization Support Environment Variables
- Repairing Garbled Fonts When Using the WE8ISO8859P15 Character Set
- Opening or Running an Encoded JSP Report
- Resolving ERR-063001 xxx.dtd null
- Running Oracle Reports in a Japanese Environment on HP-UX

Embedding a Character Set in a JSP file Dynamically

In Oracle Reports, Web report templates are configured for Western European character encoding by default. However, for other languages, you can specify the character encoding for every JSP file by using both the charset attribute of the `<Meta>` tag and the `<%@page%>` page directive.

To dynamically associate the appropriate character encoding with the JSP file, you can make the following modifications:

1. Edit the `rw*.html` files and the `blank_template.jsp` file:

   a. Modify the page directive to read

   ```
   <%@ page contentType="text/html;charset=yourIANAencoding" %>
   ```

   where:

   yourIANAencoding is the IANA-defined character set name that corresponds to the `NLS_CHARACTERSET` portion of the `NLS_LANG` variable.

   b. Modify the `<Meta>` tag inside the `<Head>` tag to read:

   ```
   <meta http-equiv="Content-Type"
   content="text/html;charset=yourIANAencoding" />
   ```

   Note: The template files; that is, `rw*.html`, and `blank_template.jsp`, are located in the *ORACLE_HOME*`/reports/templates/` directory.

2. Edit the `template.xsl` (*ORACLE_HOME*`/reports/templates/`) file:

   a. Modify the `<xsl:output>` tag to read:

   ```
   <xsl:output
        method="jsp"
        indent="yes"
        encoding="yourIANAencoding"
      />
   ```

   where

   yourIANAencoding is the IANA-defined character set name that corresponds to the `NLS_CHARACTERSET` portion of the `NLS_LANG` variable.

   b. Add the page directive to the file:

   ```
   <%@ page contentType="text/html;charset=yourIANAencoding" %>
   ```

   c. Add or modify the `<META>` tag inside the `<HEAD>` tag:

   ```
   <meta http-equiv="Content-Type"
   content="text/html;charset=yourIANAencoding" />
   ```

   where

   yourIANAencoding is the IANA-defined character set name that corresponds to the `NLS_CHARACTERSET` portion of the `NLS_LANG` variable.

Setting Globalization Support Environment Variables

The `Tk2Motif.rgb` file contains resource settings for the Motif version of the Oracle Toolkit. For example, it specifies the font mapping between the character set used by Oracle Reports, specified in `NLS_CHARACTERSET`, and X fonts.

Oracle Reports looks for this file in the directory `$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\admin\language`, where, `language` is derived from the language setting in `NLS_LANG`.

If the file does not exist, then Oracle Reports looks for the default version in $DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\admin. This version is configured for WEISO8859P1, the Western European character set.

If your NLS_LANG or NLS_CHARACTERSET specifies a character set that is not normally used for the language you have set in NLS_LANG, then Oracle Reports generates an error.

For example, if you have set NLS_LANG=AMERICAN_AMERICA.JA16EUC, then Oracle Reports locates Tk2Motif.rgb in the directory $DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\admin. The language setting in NLS_LANG is AMERICAN, and there is no language subdirectory associated with AMERICAN, so Oracle Reports uses the default file. Since this version is designed for WE8ISO8859P1, and your NLS_LANG character set is JA16EUC, Oracle Reports generates the error REP-3000.

To work around this problem, set the value of the environment variable TK_UNKNOWN to the location of your character set-specific Tk2Motif.rgb file.

For example, if NLS_LANG=AMERICAN_AMERICA.JA16EUC, then set TK_UNKNOWN=$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools\admin\JA. Even though your language is set to AMERICAN, Oracle Reports will use the Tk2Motif.rgb file in the JA language subdirectory.

With only NLS_LANG settings on Windows platform, it is not possible to display Hebrew messages in the Reports server queue. Also, by setting environment variable _JAVA_OPTIONS=-Duser.language=iw and -Duser.country=IL before starting WLS_REPORTS, the Hebrew messages can only be seen inverted in Reports Server queue page and not left-to-right.

To work around this problem:

1. Set NLS_LANG to HEBREW_ISRAEL.IW8MSWIN1255
2. Set _JAVA_OPTIONS=-Duser.language=iw -Duser.country=IL before starting WLS_REPORTS in command window.
3. Set Windows locale to Hebrew
4. Set Windows Input language to Israel/Hebrew

Repairing Garbled Fonts When Using the WE8ISO8859P15 Character Set

You may see garbled output when you run a multibyte report on UNIX with we8iso8859p15 character set.

To work around this issue, you must do the following:

1. Make an entry for the fonts used in your default printer's .ppd file. This file is located in the following directory:

   ```
   $ORACLE_HOME/guicommon/tk/admin
   ```

   Note: The entries are for the fonts that appears garbled in the report output.

2. Set the encoding scheme in the font's AFM file to FontSpecific if it is AdobeStandardEncoding. Thus, the following entry in the AFM file:

   ```
   EncodingScheme AdobeStandardEncoding
   ```

must change to:

```
EncodingScheme FontSpecific
```

Opening or Running an Encoded JSP Report

If your JSP report's character encoding (for example, EUC-JP) differs from the character set portion of the NLS_LANG environment variable (for example, JA16SJIS), then you will get the following errors:

- when running the JSP file:

```
REP-6106 or 6104 with javax.servlet.jsp.JspException  (multibyte)
REP-0495 Unable to tokenize the query (singlebyte)
```

- when opening the JSP file using Oracle Reports Builder:

```
REP-0069 Internal Error or REP-6106
```

To work around this issue, you must ensure that your JSP report's character encoding matches the IANA-defined character set corresponding to Oracle Reports' character set portion of the NLS_LANG variable.

For example:

JSP Report encoding:

```
<%@ page contentType="text/html;charset=EUC-JP" %>
<META http-equiv="Content-Type" content="text/html;charset=EUC-JP">
```

This JSP file needs to be encoded in the character set (EUC-JP).

Oracle Reports encoding:

```
NLS_LANG=JAPANESE_JAPAN.JA16EUC
```

In this example, the JSP report's encoding (EUC-JP) matches Oracle Reports' character set portion of NLS_LANG; that is, JA16EUC.

Resolving ERR-063001 xxx.dtd null

When you create a report against an XML data source, you must ensure that the encoding of both the XML file (data source) as well as the DTD matches the encoding of Oracle Reports.

When you create an XML report against a table -- for example, a Japanese table-- the group element name is in the table's language that is Japanese. To match the data source, you should set the group's element name in the DTD to Japanese. The XML and DTD files can be in any encoding that supports Japanese, for example, Shift_JIS, EUC-JP, or UTF-8. However, when the encoding of the XML data source as well as the DTD differs from Oracle Reports, you will see the following error:

```
ERR-063001 xxx.dtd null
```

Note: This error is not displayed if you use an XML schema to define the rules.

To work around this issue, you must ensure that both the data source XML files as well as the DTD file for an XML report is encoded in the character set portion of Reports Runtime NLS_LANG.

For example, if your NLS_LANG=JAPANESE_JAPAN.JA16SJIS, then both your data source XML file as well as your DTD file should be encoded in Shift_JIS.

Running Oracle Reports in a Japanese Environment on HP-UX

If you want to use Oracle Reports in the HP-UX Japanese environment with NLS_LANG=JAPANESE_JAPAN.JA16SJIS, you must modify the appropriate Tk2Motif.rgb file before using Oracle Reports because this file contains EUC encoded Japanese resources.

Convert the Tk2Motif.rgb file to Shift-JIS encoding, or remove the last seven entries from this file. Otherwise, Oracle Reports may fail.

Note: The Tk2Motif.rgb file for the Japanese environment is:

```
$DOMAIN_
HOME/config/fmwconfig/components/ReportsToolsComponent/<repo
rts_tools_name>/guicommon/tk/admin/JA.
```

Part VI

Performance

Part VI provides information on diagnosing issues and tuning your Oracle Reports Services environment:

- Chapter 24, "Diagnosing and Tuning Oracle Reports"

24

Diagnosing and Tuning Oracle Reports

As your reporting requests grow in size and complexity and your user base increases, you must 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 tune the performance of your 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 to the end result (measured in terms of the final output). See Section 24.6.1, "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 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. For more information, refer to the *Oracle Reports Services Scalability* white paper on OTN (http://www.oracle.com/technetwork/middleware/reports/collateral-9i-10g-087783.html).

This chapter will help you look at your report in the broader 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 inside firewalls)
- The degree of user interaction required

After identifying the context of your report, you can 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 objectives, you should focus your tuning on the following distinct aspects of your report:

- **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.
- **Formatting and layout**. Examine the formatting and layout of the report information.
- **Runtime parameters**. Set runtime parameters to maximize performance and distribution of reports. See Section 24.6.2, "Bursting and Distribution" for information on how distribution maximizes your reports performance.

This chapter addresses these aspects in the following sections:

24.1 Logging Enhancements

Oracle Reports provides Logging enhancements in the following areas:

- Security
- Job Administration
- Upgrade
- High Availability
- Fonts

Table 24–1 provides a list of enhanced log messages:

Table 24–1 Example enhanced logging messages.

Enhancement	Sample Output
Security	
Start	`SecurityHelper:start`
JAZNSecurity	`RWJAZNSecurity:jobCommandCheck Authorization check started`
ReportsActionHandler	`ReportsActionHandler:run Checking job command permission`
JAZNSecurity	`JAZNSecurity:jobCommandCheck Authorization check passed for job 1`
JAZNSecurity	`RWJAZNSecurity:authenticate Authenticated User weblogic successfully`
Job Administration	

Table 24–1 (Cont.) Example enhanced logging messages.

Enhancement	Sample Output
ConnectionImpl	`ConnectionImpl:runJob Calling findDuplicatedJob for jobid = 1`
JobManager	`JobManager:findDuplicatedJob Found no duplicated job for job 1`
ConnectionImpl	`ConnectionImpl:runJob No Duplicate jobs for jobid=1`
Upgrade	
Upgrade Framework	`reports app is deployed to managed server WLS_REPORTS`
Upgrade Framework	`ReportsServerComponent component is configured in destination instance`
Upgrade.Reports	`Creating reports specific upgrade items`
Upgrade.Reports	`Creating reports specific upgrade items complete.`
Font	For more information on font related log enhancements, see Section 12.1.3, "Font Diagnosis and Tracing"

24.1.1 Diagnosing Engine Crashes

This section discusses recommended settings in logging configuration to diagnose engine crashes.

Sometimes it is difficult to diagnose an engine crash as you do not find all log messages in the log file, especially when the log messages are generated just prior to an engine crash.

Due to performance reasons, the logging framework buffers the logs in memory and then flushes the logs periodically to log files. If an engine crashes suddenly, the logging framework may not get a chance to flush the logs from buffer to the file before the engine process exits. As a result, the logging information which had to be logged before an engine crash may not be available in the log file.

To diagnose an engine crash and to see all the log entries, perform the following steps:

For rwserver

- Navigate to the Reports Server `logging.xml` file located at:

  `$DOMAIN_HOME/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/logging.xml` (For Standalone Servers)

  `$DOMAIN_HOME/config/fmwconfig/servers/WLS_REPORTS/logging.xml` (For In-process Servers)

- Set the value of `autoFlushLevel` for engine log handler to `TRACE:32`.

  For example,

```
<log_handler name='rwengine_trace_handler' ....>
   <property name='autoFlushLevel' value='TRACE:32'/>
    ...
```

For rwrun

- Navigate to the `logging.xml` file located at `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/logging.xml`

- Set the value of `autoFlushLevel` for engine log handler to `TRACE : 32`.

  For example,

```
<log_handler name='runtime_trace_handler' ...>
  <property name='autoFlushLevel' value='TRACE:32'/>
   ...
```

By default, `NOTIFICATION :1` messages are flushed immediately. If you set the value of the autoFlushLevel parameter to `TRACE : 32` for any message, the log messages are flushed to the file immediately. As continuous flushing affects performance, it is recommended that you set the autoFlushLevel for diagnostic purposes only.

24.2 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 runtime 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.

This section discusses the tools you can use to monitor the performance of your report:

- Log Files
- About WLST
- Logging-Related WLST Commands
- Audit Configuration WLST Commands
- Tracing Report Execution
- RW_SERVER_JOB_QUEUE Table
- SHOWJOBS Command Line Keyword
- Efficient SQL
- PL/SQL
- Java Stored Procedures
- The Java Importer

24.2.1 Log Files

All Oracle Reports log files follow Oracle Diagnostic Logging (ODL) format, the standard across Oracle Fusion Middleware, for log format, message types, and log management directives. The log file entries are in Text format (default) or XML format. For detailed information, refer to *Administering Oracle Fusion Middleware*

Default Location of Log Files

See Table 24–2 for default location of log files.

Table 24–2 Default Location of Log Files

Component	Location, file Name

Reports standalone server	**Server log file**:
	`$DOMAIN_HOME/diagnostics/logs/ReportsServerComponent/<reports_server_name>/rwserver_diagnostic.log`
	Engine log files:
	`$DOMAIN_HOME/diagnostics/logs/ReportsServerComponent/<reports_server_name>/rwEng-<num>_diagnostic.log`
	Communication log files:
	`$DOMAIN_HOME/diagnostics/logs/ReportsServerComponent/<reports_server_name>/zrclient_diagnostic.log`

Reports In-process server and Servlet log files	**In-process server**:
	`$DOMAIN_HOME/servers/WLS_REPORTS/logs/reports/rwserver_diagnostic.log`
	I**n-process server engines**:
	`$DOMAIN_HOME/servers/WLS_REPORTS/logs/reports/rwEng-<num>_diagnostic.log`
	Servlet:
	`$DOMAIN_HOME/servers/WLS_REPORTS/logs/reports/rwservlet_diagnostic.log`

Reports Tools log files (Reports Builder, Reports Runtime, Reports Client)	**Runtime**:
	`$DOMAIN_HOME/diagnostics/logs/ReportsToolsComponent/<reports_tools_name>/runtime_diagnostic.log`
	Communication log files:
	`$DOMAIN_HOME/diagnostics/logs/ReportsToolsComponent/<reports_tools_name>/zrclient_diagnostic.log`

Reports Bridge log files	`$DOMAIN_HOME/diagnostics/logs/ReportsBridgeComponent/<reports_bridge_name>/diagnostic.log`

Tip: If you are running multiple instances of Reports Tools Components like `rwrun`, `rwbuilder`, `rwclient`, you must add the property name `keepOpen` with the value `false` to the log handler element in the logging.xml file which is present at the following location:

`$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/logging.xml`

This enables the opening and closing of the log files every time a log entry is written.

Table 24–3 12c ODL Message Types vs 10.1.2 Trace Options

ODL Message Types: Levels (12*c*)	Equivalent Trace Options (10*g* Release 2 (10.1.2))	Notes
INCIDENT_ERROR: 1	TRACE_EXC	Exceptions (unexpected internal errors)
ERROR: 1	TRACE_ERR (lower trace levels)	Errors
WARNING: 1	TRACE_WRN (lower trace levels)	Warnings
NOTIFICATION: 1	TRACE_LOG (lower trace levels)	Default level Important events for server / engine
NOTIFICATION: 16	TRACE_STA TRACE_INFO (lower trace levels)	Server / engine state info Configuration change notifications Successful / failed jobs
TRACE: 1	TRACE_PRF (lower trace levels)	Profiling information
TRACE: 8	TRACE_DST (lower trace levels)	Functional areas tracing (distribution / font handling / printing etc)
TRACE: 16	TRACE_DBG (lower trace levels)	Server - debug traces Engine - engine diagnostics

TRACE: 32	TRACE_ALL equivalent to
	{
	TRACE_APP +
	TRACE_BRK+
	TRACE_PLS +
	TRACE_SQL +
	TRACE_TMS +
	(lower trace levels)
	}
		All trace messages

The Oracle Reports log files contain the attributes listed in Table 24–4:

Table 24–4 Log File Attributes

Attribute	Description
Time stamp	Date and time when the message was generated.
Component ID	Reports messages have component ID of REP.
Message Type	The messages are categorized into the following 5 types: Error, Incident Error, Warning, Notification, Trace. (see below for more details).
Message Level	Each message is qualified by an integer value from 1 to 32 indicating the level.
Message Text	The message body.
Message ID	A unique numeric ID used in conjunction with the component ID (e.g., REP 50127). These IDs will be well documented and have proper Cause and Action associated with them.
Execution Context ID	A globally unique sequence number of the thread of execution in which the originating component participates. This is used to correlate messages from several components involved in the same thread of execution. This ID is included with all messages sent to other components. Oracle Reports generates this in case it is not passed from the originating component.
Module ID	The particular module that originated the messages. This can be any functional module in Oracle Reports (e.g., server, engine, builder).
Process ID	The operating system PID that is provided to identify the process that generated the message.
Thread ID	Identifier of the thread of execution that generated the message.

The Oracle Reports logging mechanism supports the 5 pre-defined ODL message types described in Table 24–5:

Table 24–5 ODL Message Types

Message Type	Description
Incident Error	Occurs when the program experiences an error for some internal or unexpected reason and the issue must be reported to Oracle Support.
Error	Occurs when there is any known problem that requires attention from System Administrator.
Warning	Occurs if an action occurs or a condition is discovered that should be reviewed and may require some action (else may lead to an error).
Notification	Occurs when reporting a normal action or event, such as successful login.
Trace	Refers to all the debug statements.

24.2.1.1 Viewing Log Files

You can view log files in any of the following ways:

- Using WLST Commands
- Using the Command Line

> **Note:** By default, you cannot open the log file in Microsoft Internet Explorer if the log file is in XML format. To open an XML file in Microsoft Internet Explorer, you must create a wrapper file with the top level element and include the log file(s) in it as follows:
>
> ```
> .
> <?xml version="1.0"?>
> <!DOCTYPE LOG [
> <!ENTITY log0 SYSTEM "log.xml">
> ]>
> <LOG>
> &log0;
> </LOG>
> .
> ```

Using WLST Commands

See Section 24.2.3, "Logging-Related WLST Commands".

Using the Command Line

From the command line, navigate to the following directories to open and view the log files:

- For Reports Server:

  ```
  $DOMAIN_HOME/diagnostics/logs/ReportsServerComponent/<reports_server_name>
  ```

- For Oracle Reports Bridge:

  ```
  $DOMAIN_HOME/diagnostics/logs/ReportsBridgeComponent/<reports_bridge_name>/
  ```

> **Note:** The Reports Bridge component directory is not created by default. Instead, the directory is created when a Reports Bridge is created.

- For Reports Tools components (which includes `rwrun`, `rwbuilder`, `rwclient`):

  `$DOMAIN_HOME/diagnostics/logs/ReportsToolsComponent/<reports_tools_name>/`

- For Oracle Reports Servlet (For Reports Application):

  `$DOMAIN_HOME/servers/WLS_REPORTS/logs/reports`

> **Note:** You must modify the `logging.xml` file to enable trace information for Reports components.
>
> For more information on the location of the `logging.xml` file, see Section 24.2.5, "Tracing Report Execution".

24.2.1.2 Managing Log Files

You can manage log files in Oracle Enterprise Manager, which provides capabilities such as:

- Specifying logging information.
- Searching inside log files based on various attributes of log entries.
- Viewing trend metrics; that is, how many errors of a particular type there are in log files (for example, 1 `INCIDENT_ERROR`, 20 `ERROR`, 35 `WARNING`, and so on).

See Section 6.13.1, "Specifying Logging Information" in Chapter 6, "Administering Oracle Reports Services Using Oracle Enterprise Manager".

24.2.1.3 Audit Log Files

Oracle Reports audits important events, such as the following:

- Success or failure of authentication and authorization for In-process Servers and standalone servers
- Success or failure of webcommands check based on user's role for Reports servlet

Configuring Audit Log Files

To configure audit logs, do the following:

- **For J2EE application** (In-process Server): Use Enterprise Manager to modify the audit log configuration files. See Section 6.7, "Modifying Reports Server Audit Configuration".
- For **J2SE application** (Standalone Servers): Edit the `audit.filterPreset` property in the `$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml` file as follows:

  ```
  <property name="audit.filterPreset" value="None"/>
  ```

  to

  ```
  <property name="audit.filterPreset" value="All"/>
  ```

Location of Audit Log Files

Oracle Reports audit log files are located at:

- For J2EE applications: `$DOMAIN_HOME/servers/WLS_REPORTS/logs/auditlogs/ReportsServer/audit.log`
- For J2SE components: `$DOMAIN_HOME/auditlogs/ReportsServer/<reports_server_name>/audit.log`

Audit Log Example

Following is an example for audit log:

```
2008-09-10 13:15:32.263  - "CheckAuthorization" true "Authorization of user
<username> passed for webcommand showjobs." -
"0000HlCPYyE2VOJpMkH7ie18lwWx000001,0" - - - - "12" - - - - - -
2008-09-10 13:15:25.247  - "UserLogin" true "Authentication of user <username>
passed." - "0000HlCPYyE2VOJpMkH7ie18lwWx000001,0" - - - - "12" - - - - - -
2008-09-10 13:15:25.310  - "CheckAuthorization" true "Authorization of user
<username>  passed for webcommand showjobs." -
"0000HlCPYyE2VOJpMkH7ie18lwWx000001,0" - - - - "12" - - - - - -

2008-09-10 13:15:32.263  - "CheckAuthorization" true "Authorization of user
<username> failed  for webcommand showjobs." -
"0000HlCPYyE2VOJpMkH7ie18lwWx000001,0" - - - - "12" - - - - - -
2008-09-10 13:15:25.247  - "UserLogin" true "Authentication of user <username>
failed." - "0000HlCPYyE2VOJpMkH7ie18lwWx000001,0" - - - - "12" - - - - - -
2008-09-10 13:15:25.310  - "CheckAuthorization" true "Authorization of user
<username>  failed for webcommand showjobs." -
"0000HlCPYyE2VOJpMkH7ie18lwWx000001,0" - - - - "12" - - - - - -
```

24.2.2 About WLST

The WebLogic Scripting Tool (WLST) is a command-line scripting interface, that helps you to perform administrative tasks and initiate WebLogic Server configuration changes to WebLogic Server instances and domains.

You can invoke the WLST shell by running the following command:

```
ORACLE_HOME/oracle_common/common/bin/wlst.sh
wls:/offline> connect("weblogic","weblogic", "hostname:7001")
wls:/domain2/serverConfig> domainRuntime()
```

24.2.2.1 Using WLST Commands for System Components

You can use the following WLST commands to run the Reports Components:

You can invoke the WLST shell by running the following command:

```
ORACLE_HOME/oracle_common/common/bin/wlst.sh
wls:/offline> connect("weblogic","weblogic", "hostname:7001")
wls:/domain2/serverConfig> domainRuntime()
wls:/domain2/domainRuntime> listLogs(target="ReportsServer_hostname_cinst56"))
file://hostname/scratch/rrpai/cinst56/diagnostics/logs/ReportsServerComponent/Repo
rtsServer_hostname_cinst56/server_diagnostic.log
2008-10-27 02:59:47                   302K server_diagnostic.log
file://hostname/scratch/rrpai/cinst56/diagnostics/logs/ReportsServerComponent/Repo
rtsServer_hostname_cinst56/rwEng-0_diagnostic.log
2008-10-27 02:59:13                   7.3K rwEng-0_diagnostic.log

wls:/domain2/domainRuntime> displayLogs(target="opmn:cinst56/ReportsServer_
hostname_cinst56", tail=10)

[2008-10-27T02:31:52.133-07:00] [reports] [NOTIFICATION] []
[oracle.reports.server] [host: hostname] [nwaddr: 144.20.207.149] [tid: 10] [ecid:
0000HoxdIF14EwQRyaQ5T0191Oe7000001,0] ServerConfig:logConf  <server
version="11.1.1.0.0" xmlns="http://xmlns.oracle.com/reports/server">[[

[2008-10-27T02:44:38.684-07:00] [reports] [INCIDENT_ERROR] [REP-50125]
[oracle.reports.engine] [host: hostname] [nwaddr: 144.20.207.149] [pid: 18787]
[tid: 10] [ecid: 0000HoxgDPr4EwQRyaQ5T00004_Z000000,0] [EngineName: rwEng-0]
REP-50125 : org.omg.CORBA.COMM_FAILURE:   vmcid: SUN  minor code: 201  completed:
```

```
No  [[
……

wls:/domain2/domainRuntime> displayLogs(target="opmn:cinst56/ReportsServer_
hostname_cinst56", tail=10, query="MODULE_ID equals oracle.reports.server")

[2008-10-27T02:31:01.085-07:00] [reports] [NOTIFICATION] []
[oracle.reports.server] [host: hostname] [nwaddr: 144.20.207.149] [tid: 10] [ecid:
0000Hoxd5lP4EwQRyaQ5T0191OdK000001,0] ServerConfig:logConf  <server
version="11.1.1.0.0" xmlns="http://xmlns.oracle.com/reports/server">[[

[2008-10-27T02:59:47.383-07:00] [reports] [NOTIFICATION] []
[oracle.reports.server] [host: hostname] [nwaddr: 144.20.207.149] [tid: 10] [ecid:
0000HoxjfEr4EwQRyaQ5T0191P4I000001,0] ServerConfig:logConf  Reading server config
file

wls:/domain2/domainRuntime> exit()
```

24.2.2.2 Using WLST Commands for Java EE Components

You can use the following WLST commands to run the Reports Components:

You can invoke the WLST shell by running the following command:

```
ORACLE_HOME/oracle_common/common/bin/wlst.sh
wls:/offline> connect("weblogic","weblogic", "hostname:7001")
wls:/domain2/serverConfig> domainRuntime()
wls:/domain2/domainRuntime> listLoggers(target="WLS_REPORTS")
--------------------------------------------------------------------------------
---+-----------------
Logger
| Level
--------------------------------------------------------------------------------
---+-----------------
<root>
| WARNING:1
com.sun.xml.bind.v2.ClassFactory
| <Inherited>
....
oracle.reports.adminlogconfig
| <Inherited>
oracle.reports.configMbeans
| <Inherited>
oracle.reports.engine
| NOTIFICATION:1
oracle.reports.server
| NOTIFICATION:16
oracle.reports.servlet
| NOTIFICATION:1
.....

wls:/domain2/domainRuntime> listLoggers(pattern="oracle.reports.*", target="WLS_
REPORTS")
------------------------------+-----------------
Logger                        | Level
------------------------------+-----------------
oracle.reports.adminlogconfig | <Inherited>
oracle.reports.configMbeans   | <Inherited>
oracle.reports.engine         | NOTIFICATION:1
oracle.reports.server         | NOTIFICATION:16
oracle.reports.servlet        | NOTIFICATION:1
```

```
wls:/domain2/domainRuntime> listLoggers(pattern="oracle.reports.server.*",
target="WLS_REPORTS")
----------------------+-----------------
Logger                | Level
----------------------+-----------------
oracle.reports.server | NOTIFICATION:16

wls:/domain2/domainRuntime> listLogHandlers(target="WLS_REPORTS")

Handler Name: rwservlet_handler
type:ODL
path:/scratch/rrpai/wls55/user_projects/domains/domain2/servers/WLS_
REPORTS/logs/reports/rwservlet_diagnostic.log
format:ODL-Text
maxFileSize:1M
maxLogSize:10M
Handler Name: wls-domain
type:oracle.core.ojdl.weblogic.DomainLogHandler
Handler Name: odl-handler
type:ODL
path:/scratch/rrpai/wls55/user_projects/domains/domain2/servers/WLS_
REPORTS/logs/WLS_REPORTS-diagnostic.log
maxFileSize:10M
maxLogSize:100M
```

24.2.3 Logging-Related WLST Commands

Use the following WLST commands to configure logs:

- listLoggers
- getLogLevel
- setLogLevel
- listLogHandlers
- configureLogHandlers

Use the following WLST commands to view logs:

- listLogs
- displayLogs

24.2.3.1 listLoggers

You can use the `listLoggers` command to view the list of loggers and the level of each logger.

Table 24–6 describes the parameters supported by `listLoggers` command.

Table 24–6 Parameters of listLoggers

Parameter	Description
target	The name of the WebLogic Server. The default value is the server to which WLST is connected.
pattern	A regular expression pattern that is used to filter logger names. There is no default pattern and all loggers are returned if the pattern is not provided.

Table 24–6 (Cont.) Parameters of listLoggers

Parameter	Description
runtime	A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1.

Return Value

A PyDictionary object where the keys are logger names and the associated values are the logger levels.

Examples

Following are the examples for `listLoggers` command:

1. `listLoggers()`
2. `listLoggers(pattern="oracle.*")`
3. `listLoggers(runtime=0)`
4. `listLoggers(target="server1")`

Note: ■The listLoggers command is not supported for system components, such as Reports Server and Reports Bridge.

- For more information about WLST commands, see *WLST Command Reference for WebLogic Server* guide.

24.2.3.2 getLogLevel

You can use the `getLogLevel` command to obtain the log level for a given logger.

Table 24–7 describes the parameters supported by `getLogLevel` command.

Table 24–7 Parameters of getLogLevel

Parameter	Description
target	The name of the WebLogic Server. The default value is the server to which WLST is connected
logger	A logger name.
runtime	A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1.

Return Value

The logger level as a string.

Examples

The following are the examples of getLogLevel command

1. `getLogLevel(logger="oracle")`
2. `getLogLevel(logger="oracle")`
3. `getLogLevel(logger="oracle", target="server2")`

Note: ■The getLogLevelcommand is not supported for system components, such as Reports Server and Reports Bridge

- For more information about WLST commands, see WLST Command Reference for WebLogic Server guide.

24.2.3.3 setLogLevel

You can use the `setLogLevel` command to set the log level for a given logger.

Table 24–8 describes the parameters supported by the `setLogLevel` command.

Table 24–8 Parameters of setLogLevel

parameter	Description
`target`	The name of a WebLogic server. The default value is the server to which WLST is connected
`logger`	A logger name. There is not default value.
`level`	The level name. This can be either a Java level (INFO, FINE, etc), or an ODL level (NOTIFICATION:1, TRACE:1, etc). There is no default value.
`runtime`	A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1.
`persist`	a Jython boolean value (0 or 1) that determines if the level should be saved to the configuration file. The default value is 1.

Return Value

The return value for the `setLogLevel` command is none.

Examples

The following are the examples for `setLogLevel` command.

1. `setLogLevel(logger="oracle.my.logger", level="NOTICATION:1")`
2. `setLogLevel(logger="oracle.my.logger", level="TRACE:1", persist=0)`
3. `setLogLevel(target="server1", logger="oracle.my.logger", level="WARNING", runtime=0)`

Note: ■The setLogLevel command is not supported for system components, such as Reports Server and Reports Bridge

- For more information about WLST commands, see *WLST Command Reference for WebLogic Server* guide.

24.2.3.4 listLogHandlers

You can use the `listLogHandlers` command to view the configuration of one or more log handlers.

Table 24–9 describes the parameters supported by `listLogHandlers` command.

Table 24–9 Parameters of listLogHandlers

Parameters	Description
`target`	the name of a WebLogic server. The default value is the server to which WLST is connected.
`name`	the name of a log handler. If the name is not provided then all handlers are listed.

Return Value

A java.util.List with one entry for each handlers. Each entry is a javax.management.openmbean.CompositeData object describing the handler.

Examples

The following are the examples for `listLogHandlers` command

1. `listLogHandlers()`
2. `listLogHandlers(name="odl-handler")`
3. `listLogHandlers(target="server1")`

Note: ■The listLogHandlers command is not supported for system components, such as Reports Server and Reports Bridge

- For more information about WLST commands, see *WLST Command Reference for WebLogic Server* guide.

24.2.3.5 configureLogHandlers

You can use the `configureLogHandler` command to configure and existing log handler, add a new log handler, or remove existing handlers.

Table 24–10 describes the parameters supported by `configure LogHandler` command.

Table 24–10 Parameters for configure LogHandler

Parameter	Description
`target`	The name of a WebLogic server. The default value is the server to which WLST is connected.
`name`	The name of a log handler
`maxFileSize`	The value of the maxFileSize attribute for an ODL handler. The value is a string representing a numeric value, possibly followed by a suffix indicating a size unit (k for kilobytes, m for megabytes, g for gigabytes).
`maxLogSize`	The value of the maxLogSize attribute for an ODL handler. The value is a string representing a numeric value, possibly followed by a suffix indicating a size unit (k for kilobytes, m for megabytes, g for gigabytes).

Table 24–10 (Cont.) Parameters for configure LogHandler

Parameter	Description
rotationFrequency	The value of the rotationFrequncy for an ODL handler. The value is a string representing a numeric value, possibly followed by a suffix indicating a time unit (m for minutes, h for hours, d for days). The default unit is minutes. The following special values are also accepted and are converted to a numeric value in minutes: HOUR, HOURLY, DAY, DAYLY, WEEK, WEEKLY, MONTH, MONTHLY.
baseRotationTime	the base rotation time, to be used with the rotation frequency parameter. The value must be a string representing a date/time values. It can be a full date/time in ISO 8601 date/time format, or a short form including only hours and minutes. The default baseRotationTime is 00:00.
retentionPeriod	The retention period in minutes. The value must be string representing a numeric value, possibly followed by a suffix indicating a time unit (m for minutes, h for hours, d for days). The default unit is minutes. The following special values are also accepted and are converted to a numeric value in minutes: HOUR, HOURLY, DAY, DAYLY, WEEK, WEEKLY, MONTH, MONTHLY.
format	The format for the ODL handler. The value must be the string "ODL-Text" or "ODL-XML". The default format is ODL-Text
encoding	the character encoding for the log file.
path	the log file path.
handlerType	the name of the Java class that provides the handler implementation. It must be an instance of java.util.logging.Handler or oracle.core.ojdl.logging.HandlerFactory.
propertyName	the name of a handler property to be added or update. The property value is specified with the propertyValue parameter.
propertyValue	the new value for the handler property defined by the propertyName parameter.
addProperty	a Jython boolean value. Used in conjunction with the propertyName and propertyValue parameters to define that a new property is to be added to the handler.
removeProperty	a list of one or more handler properties to be removed.
addHandler	the name of a handler to be added.
removeHandler	the name of a handler to be removed.
addToLogger	a list of logger names. The handler is added to the given logger names.
removeFromLogger	a list of logger names. The handler is removed from the given loggers.

Note: The listLogHandlers command is not supported for system components, such as Reports Server and Reports Bridge.

For more information about WLST commands, see *WLST Command Reference for WebLogic Server* guide.

24.2.3.6 listLogs

You can use the `listLogs` command to view the list of one or more components.

Table 24–11 describes the parameters supported by `listLoggers` command.

Table 24–11 Parameters of listLogs

Parameter	Description
target	The name of the WebLogic Server, or a system component. In connected mode, the default target is the WebLogic domain, and in disconnected mode there is no default.
unit	defines the unit to use for reporting file size.Valid values are B (bytes), K (kilobytes), M (megabytes), G (gigabytes), or H (display size in a human-readable form, similar to Unix's "ls -h" option The default value is H.
fulltime	a Jython Boolean value. The default value is false

Return Value

A PyArray with one element for each log. The elements of the array are javax.management.openmbean.CompositeData objects describing each log.

Examples

Following are the examples for `listLoggers` command:

1. `listLogs()`
2. `listLogs(target="server1")`
3. `listLogs(target="opmn:instance1/ohs1")`
4. `listLogs(oracleInstance="/middleware/user_projects/domains/base_domain", target="server1")`

Note:
- The listLogs command is supported for both Java EE and system components.
- For more information about WLST commands, see *Fusion Middleware WLST Command Reference Guide*.

24.2.3.7 displayLogs

You can use the `displayLogs` command to view the contents of diagnostic logs.

Table 24–12 describes the parameters supported by `listLoggers` command.

Table 24–12 Parameters of displayLogs

Parameter	Description
target	The name of the WebLogic Server, or a system component. In connected mode, the default target is the WebLogic domain, and in disconnected mode there is no default.
query	a string that specify an expression used to filter the contents of log files. A simple expression has the form "`<field-name> <operator> <value>`", where `<field-name>` is a log record field name and `<operator>` is an appropriate operator for the field type.

Return Value

The command returns a value only when the returnData parameter is set to true. By default it will not return any data. The return value depends on the option used.

Examples

Following are the examples for displayLogs command:

1. `displayLogs(tail=100)`
2. `displayLogs(target='ohs1', last=60)`
3. `displayLogs(groupBy=['COMPONENT_ID', 'MSG_TYPE'])`
4. `displayLogs(query='MSG_TYPE equals ERROR or MSG_TEXT contains Exception")`
5. `displayLogs(query='APP equals myApp', last=60)`
6. `displayLogs(query='ECID equals 0000Hl9TwKUCslT6uBi8UH18lkWX000002')`

Note:
- The listLogs command is supported for both Java EE and system components.
- For more information about WLST commands, see *Fusion Middleware WLST Command Reference Guide*.

24.2.4 Audit Configuration WLST Commands

Use the WLST commands listed in Table 24–13 to view and manage audit policies and the audit repository configuration.

Table 24–13 WLST Audit Commands

Use this command...	To...	Use with WLST...
getAuditPolicy	Display audit policy settings.	Online
setAuditPolicy	Update audit policy settings.	Online
listAuditEvents	List audit events for one or all components.	Online

24.2.4.1 getAuditPolicy

This online command displays audit policy settings including the filter preset, special users, custom events, maximum log file size, and maximum log directory size. The component mbean name is required for system components like Oracle Internet Directory and Oracle Virtual Directory.

Note: You can obtain a system component's MBean name using the `getNonJava EEAuditMBeanName` command.

The syntax of the `getAuditPolicy` command is as follows:

```
getAuditPolicy(['mbeanName'])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for system components.

Example

The following command displays the audit settings for a Java EE component:

```
wls:/domain52/serverConfig> getAuditPolicy()
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help(domainRuntime)

FilterPreset:None
Max Log File Size:104857600
Max Log Dir Size:0
```

24.2.4.2 setAuditPolicy

This online command configures the audit policy settings. You can set the filter preset, add or remove users, and add or remove custom events. The component mbean name is required for system components like Oracle Internet Directory and Oracle Virtual Directory.

Note: You can obtain a system component's MBean name using the `getNonJava EEAuditMBeanName` command.

The syntax of the `setAuditPolicy` command is as follows:

```
setAuditPolicy(['mbeanName'],['filterPreset'],['addSpecialUsers'],
['removeSpecialUsers'],['addCustomEvents'],['removeCustomEvents'])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for system components.
filterPreset	Specifies the filter preset to be changed.
addSpecialUsers	Specifies the special users to be added.
removeSpecialUsers	Specifies the special users to be removed.
addCustomEvents	Specifies the custom events to be added.
removeCustomEvents	Specifies the custom events to be removed.

Example

The following command sets audit policies to `All` level.:

```
wls:/domain52/domainRuntime> setAuditPolicy(filterPreset='All');
```

```
Already in Domain Runtime Tree
Audit Policy Information updated successfully
```

24.2.4.3 listAuditEvents

This online command displays the attributes and audit events of a component. For system components, pass the component mbean name as a parameter. Java EE applications and services like Oracle Platform Security Services (OPSS) do not need the MBean parameter. Without a component type, all generic attributes applicable to all components are displayed.

> **Note:** You can obtain a system component's MBean name using the `getNonJava EEAuditMBeanName` command.

The syntax of the `listAuditEvents` command is as follows:

```
listAuditEvents(['mbeanName'],['componentType'])
```

Argument	Definition
mbeanName	Specifies the name of the component MBean.
componentType	Specifies the component type.

Example

The following command displays all audit events:

```
wls:/domain52/domainRuntime> listAuditEvents();
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help(domainRuntime)

Components:
DIP
JPS
OIF
OWSM-AGENT
OWSM-PM-EJB
ReportsServer
WS-PolicyAttachment
WebCache
WebServices
Attributes applicable to all components:
ComponentType
InstanceId
HostId
HostNwaddr
ModuleId
ProcessId
OracleHome
HomeInstance
ECID
RID
...
```

24.2.5 Tracing Report Execution

Enabling report tracing generates a text file that describes the series of steps completed during the execution of the report. The trace file provides abundant information, which is useful not only for performance tuning but also for debugging reports and identifying performance bottlenecks.

- For Reports Builder (`rwbuilder`) and Reports Runtime (`rwrun`), specify tracing options in the `logging.xml` file.

  For example:

  In `$DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/logging.xml`, specify the trace level to `Trace 32`.

- For Reports Server (`rwserver`), specify tracing options in the logging.xml file. Separate trace files are generated for Reports Server and the engine(s).

  For example:

  – In `$DOMAIN_HOME/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/logging.xml`, specify tracing level to `Trace 32`.

  In 11*g*, job level tracing for an individual report does not exist.

- For Oracle Reports Servlet (`rwservlet`), specify tracing options in the logging xml file present at the following location:

  `FMW_HOME/user_projects/domains/<domainname>/config/fmwconfig/servers/WLS_REPORTS/logging.xml`

- You can also manually specify the tracing options in the logging.xml file present at the following location:

  `$DOMAIN_HOME/config/fmwconfig/components/ReportsBridgeComponent/<reports_bridge_name>/logging.xml`

Following is the outline of the information output to the logging file.

Example 24–1 Oracle 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 24–14 Oracle 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.

Table 24–14 (Cont.) Oracle Reports Builder

Field	Description
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 24–1, focus your tuning efforts on time formatting (Reports Time) the data rather than on querying and fetching it.

24.2.6 RW_SERVER_JOB_QUEUE Table

The `RW_SERVER_JOB_QUEUE` table provides another window (aside from that available through Enterprise Manager) into the Reports Server job queues.

The Reports Server posts information about the current report to the database when the job is enqueued and finished.

This information is inserted into the `RW_SERVER_JOB_QUEUE` table that includes the following data:

- The name of the job
- The job submitter
- The output format
- The job's current status
- When the job was queued, started, and subsequently finished

Table 24–15 lists and describes the information contained in the `RW_SERVER_JOB_QUEUE` table:

Table 24–15 Structure of the RW_SERVER_JOB_QUEUE Table

Column Name	Description
JOB_QUEUE	States whether the job listed is CURRENT, PAST, or SCHEDULED.
JOB_ID	System generated job identification number.
JOB_TYPE	Type of job, such as `report`, `rwurl`, and so on, as defined in the Reports Server configuration file, `rwserver.conf`.
JOB_NAME	Job submission name (or file name if no value for JOBNAME is specified).
STATUS_CODE	Current status of job. See Table 24–16 for more information about status codes.

Table 24–15 (Cont.) Structure of the RW_SERVER_JOB_QUEUE Table

Column Name	Description
STATUS_MESSAGE	Full message text relating to status code (includes error messages if report is terminated). See Table 24–16 for more information about status codes.
COMMAND_LINE	Complete command line submitted for this job submission.
OWNER	User who submitted the job. On the Web, the default user is the OS user who owns the Web server.
DESTYPE	Destination where report output is sent.
DESNAME	Name of the report output if not going to the Reports Server cache.
SERVER	Reports Server to which the report was submitted.
QUEUED	Date and time the job submission was received and queued by the given Reports Server.
STARTED	Date and time the job submission was run.
FINISHED	Date and time the submitted job completed.
RUN_ELAPSE	Elapsed time between started and finished time, in units of milliseconds.
TOTAL_ELAPSE	Elapsed time between queued and finished time, in units of milliseconds.
LAST_RUN	Date and time a scheduled job was last run.
NEXT_RUN	Date and time a scheduled job will run.
REPEAT_INTERVAL	Frequency on which to run a job.
REPEAT_PATTERN	Repeat pattern (for example, every minute, every hour, or every day).
CACHE_KEY	Cache key used to compare a request with an already cached result. The key is a string that uniquely indicates a report output result without considering the time the job was run. For example, if two requests have the same key, it means they will both generate the same output if they are running at the same time, although the outputs may be used for different purposes (for example, sent to e-mail or saved to a file).
CACHE_HIT	Indicates whether the job result was fetched from cache instead of running itself.

Table 24–16 Job Submission Status Codes

Status Code	Defined PL/SQL Constant	Description for Status Code
1	ENQUEUED	Job is waiting in queue.
3	RUNNING	Report is currently running.
4	FINISHED	Job submission has completed successfully. This code will not change once set.
5	TERMINATED_W_ERR	Job has ended with an error.
7	CANCELED	Job was canceled by user request.
8	SERVER_SHUTDOWN	Job was canceled due the Reports Server shutting down.

Table 24–16 (Cont.) Job Submission Status Codes

Status Code	Defined PL/SQL Constant	Description for Status Code
11	TRANSFERRED	Job is transferred to another server in the cluster (note: clustering was deprecated in 10*g* Release 2 (10.1.2)). This code will not change once set. In this case, the job is submitted to another Reports Server as a "new" job (so that the user can query the new Reports Server for the new job's status).
12	VOID_FINISHED	Job is finished but output is void because of reaching limit of cache capacity.
13	ERROR_FINISHED	Output is successfully generated but failed to send to destinations.
15	EXPIRED	Scheduled report has expired.

Users can view this table if you grant them SELECT access. This will enable them to query the job submission of interest and determine the job's current status. You can also give them a view of this data by implementing a Reports Server Queue screen. You can implement such a screen by creating a report based directly on this table. Doing so displays the queue report as a job submission by the user.

Conversely, the real-time update of the table with the status of job submissions makes it very easy for administrators to know exactly how many concurrent users have requested jobs to be run on the Reports Server.

By counting the number of entries in the `RW_SERVER_JOB_QUEUE` table that have a status code indicating that the job has been queued but not completed, it is possible to return an accurate number of the current active users on the server. For example, you could use the following query:

```
SELECT Count(*)
FROM   RW_SERVER_JOB_QUEUE
WHERE  STATUS_CODE IN (1,     -- ENQUEUED
                       2,     -- OPENING
                       3)     -- RUNNING
AND    JOB_TYPE != 'Scheduled'
```

> **Note:** While the table contains the date and time a report was queued, run, and finished, it is not a good idea to use a query based on the fact that a job has a defined `QUEUED` and `STARTED` time but no `FINISHED` value. If a report ends due to an unexpected error, such as invalid input, then the `FINISHED` column remains `NULL`. However, the `STATUS_CODE` and `STATUS_MESSAGE` both indicate there has been a failure and list the cause of that failure.

24.2.6.1 Updating the Database with Queue Activity

The Reports Server job queue is implemented through the use of a PL/SQL case API. It functions to update the queue table with the queue information as requests are made. This implementation is defined in the following path:

```
ORACLE_HOME\reports\admin\sql\rw_server.sql
```

This script is certified to worked against Oracle 10*g* database.

To implement the queue, perform the following steps:

1. Load the rw_server.sql file to a database (this file is included with your Oracle Reports Services installation: *ORACLE_HOME*\reports\admin\sql).

   This creates a schema that owns the report queue information and has execute privileges on the server queue API. For backward compatibility with Oracle6*i* Reports, this also creates a view called RW_SERVER_QUEUE.

2. Set the repositoryconn property of the jobStatusRepository element in the server configuration file (${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/rwserver.conf for Standalone servers and $DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/rwserver.conf for In-process servers) to the connection string of the schema that owns the queue data. See Section 7.2.1.11, "jobStatusRepository".

   Alternatively, you can set the jobstatusRepository in Oracle Enterprise Manager.

When the server starts, it connects as the defined user and logs job submissions.

Note: Oracle Reports uses the dbconn property of the jobstatusrepository element to connect to the database when updating the log information about job queues

24.2.7 SHOWJOBS Command Line Keyword

You can use SHOWJOBS on the command line to display a Web view of Reports Server queue status for reports run through rwservlet.

See Section A.8.8, "SHOWJOBS".

24.2.8 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 Oracle Reports Builder. However, the wizard cannot prevent inefficient SQL from being created, such as SQL that does not use available indexes.

To tune your report SQL, use the trace functionality available in the Oracle database. SQL tracing enables 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 Oracle 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 trigger, or a Before Parameter Form trigger.

The following EXPLAIN PLAN map was generated using the database's SQL trace facility. Refer to the Oracle Database PL/SQL Language Reference 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; that is, 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; that is, 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 Fusion Middleware documentation provides more information on the database optimizer's functionality.

24.2.9 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.

Note: For a description of the ORA built-in package see the *Oracle Reports online Help*.

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 Oracle 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.

Note: For a description of the SRW built-in package, including the SRW.DO_SQL built-in procedure, see the *Oracle Reports online Help*.

Example

```
SRW.DO_SQL (`CREATE TABLE' || :tname ||`(ACCOUNT NUMBER
  NOT NULL PRIMARY KEY, COMP NUMBER (10,2))');
```

24.2.10 Java Stored Procedures

Java stored procedures enable you to implement business logic at the server level; thereby, improving application performance, scalability, and security. Oracle Database 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 *Oracle Database Java Developer's Guide* for more information on Java stored procedures.

24.2.11 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 enables 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.

24.3 Tuning Reports Server Configuration

This section provides tips for improving the performance and stability of Reports Server, which is responsible for:

- Accepting the report request from various clients.
- Scheduling the jobs to run.
- Managing Oracle Reports engines
- Managing the cache
- Managing various destinations
- Security check
- Managing the jobstore (persistent job data)

While operating under heavy load, it is essential to tune various Reports Server parameters to optimal values, as follows:

1. Determine optimal values for the `initEngine`, `maxEngine`, and `minEngine` attributes of the `engine` element in the server configuration file:

   ```
   <engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="1"
   maxEngine="2" minEngine="1" engLife="50"
   maxIdle="30" callbackTimeOut="90000">
   ```

   For more information on the `engine` element, refer to Section 7.2.1.8, "engine". The `maxEngine` value sets the maximum number of processes ready to respond to user requests for running reports. Setting it too low means user requests get queued up and available machine capacity is not fully utilized. Setting it too high means Reports Server will take more than its share of machine capacity from other activities the host also needs to perform, and could cause the operating system to crash.

   As an example of a simple calculation for number of engines, suppose you have set of reports that takes an average of 10 seconds to run. Input requests to your system varies from 6 reports per minute to 12 reports per minute. In this scenario, the calculations are as follows:

   - `initEngine` = (average time to run report) * (minimum report requests input rate) = (10/60) * 6 = 1
   - `maxEngine` = (average time to run report) * (maximum report requests input rate) = (10/60) * 12 = 2
   - `minEngine` = Depending on the kind of load, anything between 0 to `initEngine`

   With these calculations, `minEngine=1` and `maxEngine=2` can be specified in the server configuration file. This ensures that whenever a job arrives, it gets an idle engine immediately.

   In scalability and performance tests, maximum throughput is seen when `maxEngine` is configured using the guideline of 2-4 engines multiplied by the number of CPUs. You can set up a maximum value of maxEngine=8 if the machine has a dual processor or maxEngine=16 if the machine has a quad processor.

   If you are not using the URL engine, comment the `engine` element with `ID="rwURLEng"` in the server configuration file.

2. Determine optimal values for the `cache` element's `cacheSize` property, the `queue` element's `maxQueueSize` attribute, and the `EXPIRATION` keyword.

   For more information, refer to Section 7.2.1.3, "cache", Section 7.2.1.20, "queue", and Section A.6.6, "EXPIRATION". The values of `cacheSize`, `maxQueueSize`, and `EXPIRATION` are related to each other and they must be set carefully for efficient Reports Server operation.

   For example, when you run reports with `EXPIRATION=480`, this implies that you want to keep the jobs in cache for 4 hours (480 minutes). Given that, `maxQueueSize` should be set to accommodate all the jobs for 4 hours. Thus, at a rate of 10 jobs per minute:

   `maxQueueSize` = (report requests input rate) * (expiration period) = 480 * 10 =4800.

   The value of `cacheSize` also should be set sufficiently high to accommodate 4800 jobs. Suppose the average size of each report is 100K:

   `cacheSize` = (`maxQueueSize`) * (average size of report) = 4800*100/1000 = 480MB

   You can use similar logic to calculate the value of the `cache` element's `maxCacheFileNumber` property.

> **Note:** The minimum recommended value for `maxQueueSize` is 1000 (the default). A significantly lower value than the default values for `maxQueueSize` or `cacheSize` may degrade Reports Server performance.

3. Set the `engineResponseTimeOut` attribute of the `engine` element in the server configuration file:

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl" initEngine="1"
maxEngine="2" minEngine="1" engLife="50" maxIdle="30"
callbackTimeOut="90000" engineResponseTimeOut="5">
```

   For more information on the `engine` element, refer to Section 7.2.1.8, "engine".

   Set `engineResponseTimeOut` if you are experiencing intermittent engine hangs. This attribute enables Reports Server to detect the hanging engine and perform cleanup. The sooner Reports Server detects the hang, the better the stability of the system. Thus, `engineResponseTimeOut` must be set carefully, as follows:

   The value of `engineResponseTimeOut` should be set to the maximum time a report takes in the set of reports you have. For example, if you have set of reports that takes 10 seconds to 5 minutes to run, you can set `engineResponseTimeOut="5"` (5 minutes).

   > **Note:** It is always better to run batch reports on a separate server with different `engineResponseTimeOut` values. Do not submit interactive and batch reports to same server.

4. Set the `maxConnect` attribute of the `connection` element in the server configuration file;

```
<connection maxConnect="180" idleTimeOut="15">
```

   For more information on the `connection` element, refer to Section 7.2.1.4, "connection".

   The `maxConnect` attribute controls how many total requests Reports Server can simultaneously handle at any moment in time. The key purpose of `maxConnect` is to keep Reports Server from being overcome by some runaway program or process or by a denial of service attack. It should be always set to a value that is greater than the maximum simultaneous clients.

   For example, if your system is expected to handle 150 simultaneous clients, you can set `maxConnect` to any value above 150. You can use a safety factor of 10% to 20%, as follows:

   `maxConnect` = 150 + 150 * 0.2 = 180

5. Set the HTTP timeout value (applicable to AS only).

   The HTTP timeout value should be set based on the time required to run the longest report in the system. If longest-running report takes 20 minutes to run, HTTP timeout should be more than 20 minutes. Otherwise, an HTTP timeout error will display when the report is still running in the server. This parameter can be set in the *`$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/httpd.conf`* file.

24.4 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 must 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.

This section discusses ways to review and improve the efficiency of the data used in your report:

- Non-SQL Data Sources
- Database Indexes
- Calculations
- Redundant Data
- Break Groups
- Group Filters
- To Link or Not To Link

24.4.1 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 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 ensure 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 column wise.

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 want 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. See Chapter 14, "Configuring and Using the Pluggable Data sources".

24.4.2 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.

24.4.3 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 Oracle Database 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 Oracle 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
```

24.4.4 Redundant Data

A report's query should ideally select only required columns. The fewer queries you have, the faster your report will run. Single-query data models execute more quickly than multiquery 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.

> **Note:** For a description of the `SRW` built-in package, including the `SRW.SET_MAXROW` built-in procedure, see the *Oracle Reports online Help*.

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 (that is, 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.

24.4.5 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 less 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 in spite 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.

24.4.6 Group Filters

Group filters reduce the number of records displayed. Filtering takes place after the query returns the data (from the data source) to Oracle Reports. Even if the filter is defined to display only the top five records, the result set 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.

24.4.7 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 multitable 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 must decide when to achieve the balance between performance and maintenance requirements.

24.5 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 or control the rendering mechanism. It merely *injects* data into a regular JSP page.

This section discusses reviewing and tuning the format of your report:

- Paper Layout (including Format Triggers and Image Outputs)
- Web Layout and JSP Report Definition

24.5.1 Paper Layout

When generating a default paper layout, Oracle Reports places a frame around virtually every object to prevent the object from being overwritten by another object. At runtime, every layout object (frames, fields, boilerplate, and so on) 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.

An object that is defined as variable, expanding, or contracting in either or both the horizontal or vertical directions requires extra processing. 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 Oracle 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, Oracle 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.

24.5.1.1 Format Triggers

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:

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 this example, had the format trigger contained an INSERT statement, then two rows of data would have been inserted.

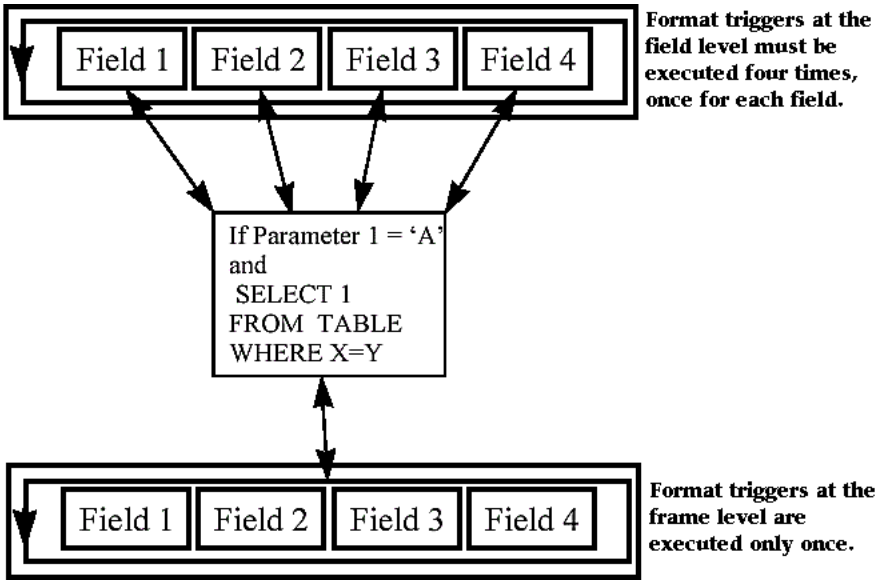
Format triggers can be used against repeating frames to filter data. However, by introducing filtering at appropriate levels, you not only improve a report's performance but also reduce the complexity required for this type of a report.

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 24–1 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 *Oracle Database PL/SQL Language 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.

24.5.1.2 Image Outputs

You can improve the performance of reports that include images by judiciously setting environment variables related to image support.

Improving performance of graphs output to a PDF file or a printer

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.

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 72 DPI. The PDF file size also increases 5 to 6 times.

This functionality is currently not supported in Oracle Reports distribution functionality, as this is specific to PDF and printer outputs only.

Note: When you set a DPI value greater than 250 and your graph is bigger than 5"x5" (approximately), you may also want to change the JVM heap size value using the REPORTS_JVM_OPTIONS environment variable to avoid the Out Of Memory error for the JVM.

For more information, refer to Section B.1.51, "REPORTS_GRAPH_IMAGE_DPI".

Improving performance of JPEG/GIF/PNG output image formats

If your input image format is JPEG, it is recommended that you do not set the REPORTS_OUTPUTIMAGEFORMAT environment variable to GIF or PNG, which will increase the image size more and might degrade the performance problem. Similarly, if your input image format is GIF or PNG, it is recommended that you do not set the REPORTS_OUTPUTIMAGEFORMAT environment variable to JPEG. For better performance, use the same format for both input and output format.

For more information, refer to Section B.1.60, "REPORTS_OUTPUTIMAGEFORMAT".

Improving performance of JPEG images

The REPORTS_JPEG_QUALITY_FACTOR environment variable specifies the level of image quality desired for JPEG images. It provides control over the trade-off between JPEG image quality and size of the image. The better the quality of the image, the greater the image file size and lower performance. If you want to improve the performance, set value to 0. The default value is 100 (highest quality). A value of 75 provides a good quality image, while ensuring a good compression ratio.

For more information, refer to Section B.1.54, "REPORTS_JPEG_QUALITY_FACTOR".

24.5.2 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 Oracle 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 Oracle Database 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 (for example, 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 keyword to prevent Oracle Reports executing the paper layout formatting.

24.6 General Layout Guidelines

This section outlines guidelines that you can follow when designing your report's layout to improve performance:

- Fetching Ahead
- Bursting and Distribution

24.6.1 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.

24.6.2 Bursting and Distribution

With 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 enables 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
```

24.7 Running the Report

You can further affect the overall performance by setting specific runtime options:

- Oracle 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, Oracle 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 Oracle Reports Services is off. To turn this option on, specify `VALIDATETAG=YES` in your HTTP request (for example, `http://my.server.com/myreport.jsp?VALIDATETAG=YES`).

  Note: Using `VALIDATETAG=YES` slows performance.

- By default, the `RECURSIVE_LOAD` command line keyword 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=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 keyword (defined in kilobytes) to a large value.
- As discussed in Section 24.3, "Tuning Reports Server Configuration", when running a large number of reports with Oracle Reports Servlet (`rwservlet`) or Reports Client (`rwclient`), set the `EXPIRATION` command line keyword to reflect the `maxQueueSize` and `cacheSize` values. For example, if the `queue` element in the

server configuration files specifies maxQueueSize=6000, you can keep a maximum of 6000 jobs in the job queue. If you run more than 6000 jobs within a day, with EXPIRATION=1440 (1 day), you may lose some of the jobs even before the EXPIRATION time is met because Reports Server will remove the jobs to maintain the maxQueueSize and server stability, even though the jobs have not expired. Additionally, the cache element in the server configuration file should specify sufficient cacheSize should be allocated in order to maintain the 6000 jobs. As a general guideline, set EXPIRATION, maxQueueSize, and cacheSize according to the number of jobs you will run in one day.

- Set the LONGCHUNK command line keyword 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 Oracle database server, use the more efficient CLOB or BLOB data types, instead of LONG or LONG RAW.
- If the Paper Parameter Form is not required, set the PARAMFORM command line keyword to NO.
- Use the COPIES command line keyword 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.
- When generating a report to PDF output, set the PDFCOMP command line keyword to NO. PDF output is compressed by default. Although compressed files download quickly, the time taken to generate a compressed file is much more when compared to a non-compressed file.

Note: Running a Report to WebLayout works only if the weblayout port in the ${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/rwbuilder.conf file is set to the port of the managed server, as in the following example:

```
 <webLayout port="managed_server_port"
docroot="$DOMAIN_HOME/servers/WLS_REPORTS/tmp/_WL_user/reports_
<version>/<random_string>/war"/>
```

Part VII

Appendixes

Part VII contains appendixes that provide additional, detailed information about functioning in Oracle Reports Builder and in the Oracle 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 Oracle Portal using batch scripts:

A

Command-Line Keywords

This appendix contains descriptions and examples of command-line keywords that can be used with the Oracle Reports components.

> **Note:** For examples of using command line keywords in your runtime URL, see Chapter 18, "Running Report Requests".

This appendix contains the following topics:

- Using the Command Line
- Overview of Oracle Reports Components
- Keyword Usage Summary
- Command-Line Keywords

The information in this appendix is also documented in the *Oracle Reports online Help*, which is available in Reports Builder or hosted on the Oracle Technology Network (OTN), as described in the Preface under "Related Documentation".

A.1 Using the Command Line

An Oracle Reports command on the command line generally has the following form:

```
component_name keyword=value, keyword=value, ...
```

where each `keyword=value` pair is called a *command line option*.

Keywords must be specified and can be used in any order following the component name.

A.1.1 General Usage Notes

- No spaces should be placed before or after the equal sign of an option.
- Separate options with one or more spaces; do not use commas to separate options.
- Values may be in single or double quotes. The effect of single or double quotes is operating system-specific.
- The `keyword=` part of all options is not case-sensitive. The `value` portion may be case-sensitive, depending on your operating system.
- To pass a single quote from the command line, you must enter two quotes (one quote as an escape and one as the actual quote). For example:

```
rwrun REPORT=myrep DESTYPE=file DESNAME=run.out BATCH=yes p_value="Roy''s Batch
Report"
```

- Full pathnames are supported for all file references (for example, DESNAME=/revenues/q1/nwsales). If you do not specify the full path name, the Oracle Reports file searching method is used to find the file. If you do not 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.
- 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 following command line keywords:
  - CMDFILE=*filename*. In this case, the CMDFILE specified is read and appended to the original command line (of which CMDFILE is a part) before being sent to the Reports Server. The runtime engine does not reread the command file
  - DESNAME=*filename* DESTYPE=LOCALFILE. In this case, DESNAME refers to files on the client machine.

A.1.2 Rules

- Values entered on the Runtime Parameter Form override those entered on the command line. For example, if you specify rwrun on the command line with COPIES=1, but in the Runtime Parameter Form, specify COPIES=2, then two copies of the report are generated.
- Values entered on the command line override those specified in command files. For example, if you specify rwrun on the command line with COPIES=1 and CMDFILE=RUNONE (a command file), but the command file RUNONE, includes rwrun COPIES=2, only one copy of the report is generated.
- You can specify values for DESTYPE, DESNAME, DESFORMAT, ORIENTATION, and COPIES in a number of different places. The following list shows the decreasing order of precedence for the places where you specify these values:
  1. Print Job dialog box
  2. Runtime Parameter Form
  3. Runtime Parameters/Settings tab of Preferences dialog box
  4. Keywords on the command line
  5. Values specified in the report definition
  6. Choose Printer dialog box

A.2 Overview of Oracle Reports Components

This section provides a brief description of the Oracle Reports components and contains examples showing how to use command-line keywords with each component.

- rwclient
- rwrun
- rwbuilder
- rwconverter
- rwservlet

- rwserver
- rwbridge

A.2.1 rwclient

`rwclient` (Reports Client) parses and transfers a command line to the specified Reports Server.

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 following command line keywords:

- `CMDFILE=`*`filename`*. In this case, the CMDFILE specified is read and appended to the original command line (of which `CMDFILE` is a part) before being sent to the Reports Server. The runtime engine does not reread the command file
- `DESNAME=`*`filename`* `DESTYPE=LOCALFILE`. In this case, `DESNAME` refers to files on the client machine.

Refer to Table A–1 for the keywords that can be used with `rwclient`.

Examples

Example 1: Running a paper report to cache

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb
desformat=pdf DESTYPE=cache
```

Example 2: Sending 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 report output to a printer

```
rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb
DESTYPE=printer DESNAME=myprinter
```

Example 4: Sending report output through 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 report output to WebDAV (any WebDAV server or Oracle 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 report output to Oracle 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 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 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 Mar 25, 2009 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"
```

A.2.2 rwrun

`rwrun` (Reports Runtime) runs a report by starting its own in-process server (not to be confused with the default in-process Reports Server), which runs in the same JVM as the `rwrun` process.

The configuration file for this in-process server is `rwbuilder.conf` and trace files are saved in the `rep_`*`machinename`*`-rwbuilder` directory.

> **Note:** It is recommended that you use `rwrun` for testing purposes only. Use `rwservlet` and `rwclient` in your production environment to take full advantage of the power of Oracle Reports Services.

Refer to Table A–1 for the keywords that can be used with `rwrun`.

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: Sending report output to a file

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf DESTYPE=file
DESNAME=c:\mydir\test.pdf
```

Example 3: Sending report output to a printer

```
rwrun report=test.rdf userid=scott/tiger@mydb DESTYPE=printer
DESNAME=myprinter
```

Example 4: Sending report output through 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 5: Sending report output to WebDAV (any WebDAV server or Oracle Portal WebDAV)

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=htmlcss
DESTYPE=webdav
"DESNAME"="http://myusername:mypassword@mywebdavserv.com/mydir/test.html
"
```

Example 6: Sending report output to Oracle Portal

```
rwrun report=test.rdf userid=scott/tiger@mydb DESTYPE=oracleportal
desformat=PDF PAGEGROUP=mypagegrp OUTPUTPAGE=reports_output
ITEMTITLE=pushtoportal STATUSPAGE=result
```

Example 7: Sending XML PDS report output to a file

```
rwrun report=myxmlpdstest.rdf destype=file desformat=PDF
desname=c:\mydir\my.pdf
```

Example 8: Sending 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 9: Distributing report output to multiple destinations

```
rwrun report=test.rdf userid=scott/tiger@mydb DISTRIBUTE=yes
DESTINATION=c:\mydistribute.xml
```

Example 10: Using a secured Reports Server

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf destype=file
desname=test.pdf AUTHID=myadmin/myadmin
```

Example 11: 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"
```

A.2.3 rwbuilder

`rwbuilder` invokes Oracle Reports Builder. When you include a `REPORT|MODULE` keyword on the command line with `rwbuilder`, Oracle Reports Builder opens with the specified report highlighted in the Object Navigator. When no report is specified, Oracle Reports Builder opens with a Welcome dialog offering you the choice of opening an existing report or creating a new one.

Refer to Table A–1 for the keywords that can be used with `rwbuilder`.

Example

```
rwbuilder report=myrep.rdf userid=scott/tiger@mydb
```

A.2.4 rwconverter

`rwconverter` (Reports Converter) 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. Currently, Reports does not support the use of .PLX files.

- Create a PL/SQL script that batch registers reports in Oracle Portal

In some cases, `rwconverter` automatically compiles 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. If there are compile errors, `rwconverter` displays a warning, but the conversion continues.
- Using a report created on another platform than the source. If there are compile errors, `rwconverter` displays a warning, but the conversion continues.

In all other situations, you must compile the report's PL/SQL yourself (for example, using **Program > Compile > All** in Oracle Reports Builder).

Note: Fonts are mapped when a report is opened by Oracle Reports Builder or Reports Runtime, not during the conversion.

Refer to Table A–1 for the keywords that can be used with `rwconverter`.

Example:

```
rwconverter scott/tiger@mydb stype=rdffile source=inven1.rdf dtype=xmlfile
dest=inven1_new.xml
```

A.2.5 rwservlet

`rwservlet` (Oracle Reports Servlet) translates and delivers information between either a Web server or a Java EE Container (for example, Oracle WebLogic Server) and the Reports Server, allowing you to run a report dynamically from your Web browser. Optionally, it can use the in-process Reports 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.

The keywords used with `rwservlet` are also known as *Web commands*. Oracle Reports provides security access levels for `rwservlet` Web commands, based on two categories:

- End user Web commands are: GETJOBID, KILLJOBID, SHOWAUTH, SHOWJOBID.
- Administrator-only Web commands are: DELAUTH, GETSERVERINFO, KILLENGINE, PARSEQUERY, SHOWENV, SHOWJOBS, SHOWMAP, SHOWMYJOBS.

  Administrators are allowed to run both end user and administrator-only Web commands. For a non-secured server, the user ID and password for administrators can be set in the identifier element of the Reports Server configuration file.

The security access levels are specified using Oracle Enterprise Manager, as described in Section 6.3.4, "Defining Security Policies for Web Commands":

Note: When you use `rwservlet` to run a JSP, you can use all keywords applicable to `rwservlet`. For more information on running a JSP with `rwservlet`, see Section 18, "Running Report Requests".

Note: The following keywords are commands rather than keyword=value pairs; that is, these keywords are entered by themselves without a corresponding value: SHOWENV, SHOWJOBS, SHOWMAP, SHOWMYJOBS, KILLJOBID, KILLENGINE, PARSEQUERY, DELAUTH, GETJOBID, and GETSERVERINFO.

Refer to Table A–1 for the keywords that can be used with `rwservlet`.

Examples

In the following examples, `myias.mycomp.com` is your Oracle Application Server instance, and `7779` is the port where `rwservlet` is running.

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 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 report output to a printer

```
http://myias.mycomp.com:7779/reports/rwservlet?
server=myrepserv+report=test.rdf+userid=scott/tiger@mydb+DESTYPE=printer
+DESNAME=myprinter
```

Example 4: Sending report output to 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@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 Oracle Portal WebDAV)

```
http://myias.mycomp.com:7779/reports/rwservlet?
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 report output to Oracle Portal

```
http://myias.mycomp.com:7779/reports/rwservlet?
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 XML PDS report output to a file

```
http://myias.mycomp.com:7779/reports/rwservlet?
server=myrepserv+report=myxmlpdstest.rdf+destype=file+
desformat=PDF+DESNAME=c:\mydir\my.pdf
```

Example 8: Sending JDBC PDS report output to a file

```
http://myias.mycomp.com:7779/reports/rwservlet?
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 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.rdf+
SCHEDULE="every first fri of month from 15:53 Oct 23, 2007 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=myadmin/myadmin
```

Example 12: Using a key map file (see Section 18.14, "Using a Key Map File")

```
http://myias.mycomp.com:7779/reports/rwservlet?key1
```

where

`key1` is a key defined in the `cgicmd.dat` file (the keyname should be the first parameter)

or

```
http://myias.mycomp.com:7779/reports/rwservlet?
server=myrepserv+userparam=12+CMDKEY=keyname
```

Example 13: Running a report with a Parameter Form

```
http://myias.mycomp.com:7779/reports/rwservlet?
server=myrepserver+report=test.rdf+userid=scott/tiger@mydb+
destype=cache+desformat=htmlcss+PARAMFORM=html
```

Example 14: Running a report with e-mail notification

```
http://myias.mycomp.com:7779/reports/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/reports/rwservlet?
server=myrepserver+report=test.rdf+userid=scott/tiger@mydb+
destype=file+desformat=pdf+desname=test.pdf+JOBTYPE=rwurl+
URLPARAMETER="http://www.oracle.com"
```

Example 16: Showing the environment information for server `myrepserver`

```
http://myias.mycomp.com:7779/reports/rwservlet/SHOWENV?
server=myrepserver+authid=myrepuser/myreppassword
```

Example 17: Viewing the past jobs information for server `myrepserver`

```
http://myias.mycomp.com:7779/reports/rwservlet/SHOWJOBS?server=myrepserv
er+authid=myrepuser/myreppassword+queuetype=past
```

Example 18: Viewing the `cgicmd.dat` key mappings (see Section 18.14, "Using a Key Map File")

```
http://myias.mycomp.com:7779/reports/rwservlet/SHOWMAP?authid=myrepuse
r/myreppassword
```

Example 19: Viewing current jobs information for user `myrepuser`

```
http://myias.mycomp.com:7779/reports/rwservlet/SHOWMYJOBS?
server=myrepserver+authid=myrepuser/myreppassword+
queuetype=current
```

Example 20: Getting the status of a job with job ID `30`

```
http://myias.mycomp.com:7779/reports/rwservlet/SHOWJOBID30?
server=myrepserver+authid=myrepuser/myreppassword
```

Example 21: Cancelling a currently running job with job ID `122`

```
http://myias.mycomp.com:7779/reports/rwservlet/KILLJOBID122?
server=myrepserver+authid=myrepuser/myreppassword
```

Example 22: Viewing the parsed query of a command

```
http://myias.mycomp.com:7779/reports/rwservlet/PARSEQUERY?
server=myrepserver+authid=myrepuser/myreppassword+
report=test.rdf+userid=scott/tiger@db+destype=cache+
desformat=htmlcss
```

Example 23: Showing DB authentication page

```
http://myias.mycomp.com:7779/reports/rwservlet/SHOWAUTH?
server=myrepserver+authid=myrepuser/myreppassword+authtype=D
```

Example 24: Deleting cookies set by `rwservlet`

```
http://myias.mycomp.com:7779/reports/rwservlet/DELAUTH?
server=myrepserver+authid=myrepuser/myreppassword
```

Example 25: Getting the output of job with job ID `87` from server `myrepserver`

```
http://myias.mycomp.com:7779/reports/rwservlet/GETJOBID87?
server=myrepserver+authid=myrepuser/myreppassword
```

Example 26: Displaying server information for server `myrepserver`

```
http://myias.mycomp.com:7779/reports/rwservlet/GETSERVERINFO?
server=myrepserver+authid=myrepuser/myreppassword
```

Example 27: Killing engine `rwEng-1` in server `myrepserver`

```
http://myias.mycomp.com:7779/reports/rwservlet/KILLENGINE1?
type=rwEng+server=myrepserver+authid=myrepuser/myreppassword
```

A.2.6 rwserver

`rwserver` (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 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.

Refer to Table A–1 for the keywords that can be used with `rwserver`.

For manual configuration, see Section 7.2, "Reports Server Configuration File".

A.2.7 rwbridge

`rwbridge` (Oracle Reports Bridge) is used when Reports Server and Oracle Reports Client are in different Farms. Oracle Reports Client uses the default broadcast mechanism for server discovery, which sends packets that can travel only within a Farm. The Oracle Reports Bridge can bridge two Farms in a network. It intercepts the packets broadcast by Reports Server and Oracle Reports Client and transfers them to the remote bridges configured in the Oracle Reports Bridge configuration file.

There are no command line keywords for `rwbridge`.

To configure the Oracle Reports Bridge, refer to Section 6.1, "Configuring Oracle Reports Components".

To start and stop the Oracle Reports Bridge, see Chapter 5, "Starting and Stopping Oracle Reports Services".

A.3 Keyword Usage Summary

Table A–1 provides an alphabetical summary list of all the Oracle Reports command line keywords and specifies the Oracle Reports components with which each keyword can be used.

An asterisk (*) after a keyword indicates that the keyword is maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

Table A–1 Keywords Usage Summary

Keywords	rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver	rwbridge
ACCESSIBLE	yes	yes	yes	no	yes	no	no
ARRAYSIZE	yes	yes	yes	no	yes	no	no
AUTHID	yes	yes	no	no	yes	yes	no
AUTOCOMMIT	yes	yes	yes	no	yes	no	no
BACKGROUND	yes	no	no	no	yes	no	no
BATCH	no	no	no	yes	no	yes	no
BCC	yes	yes	no	no	yes	no	no
BLANKPAGES	yes	yes	yes	no	yes	no	no
BUFFERS	yes	yes	yes	no	yes	no	no
CACHELOB	yes	yes	yes	no	yes	no	no
CC	yes	yes	no	no	yes	no	no
CELLWRAPPER	yes	yes	no	no	yes	no	no
CMDFILE	yes	yes	yes	yes	no	no	no
CMDKEY	no	no	no	no	yes	no	no
COLLATE	yes	yes	no	no	yes	no	no
COMPILE_ALL	no	no	no	yes	no	no	no
CONTAINSHTMLTAGS	yes	yes	yes	no	yes	no	no
CONTAINSOLE	yes	yes	yes	no	yes	no	no
CONTENTAREA*	yes	yes	no	no	yes	no	no
COPIES	yes	yes	no	no	yes	no	no
CUSTOMIZE	yes	yes	no	yes	yes	no	no

Table A–1 (Cont.) Keywords Usage Summary

Keywords	rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver	rwbridge
DATEFORMATMASK	yes	yes	no	no	yes	no	no
DBPROXYCONN	yes	yes	no	no	yes	no	no
DELAUTH	no	no	no	no	yes	no	no
DELIMITED_HDR	yes	yes	no	no	yes	no	no
DELIMITER	yes	yes	no	no	yes	no	no
DESFORMAT	yes	yes	no	no	yes	no	no
DESNAME	yes	yes	no	no	yes	no	no
DEST	no	no	no	yes	no	no	no
DESTINATION	yes	yes	no	no	yes	no	no
DESTYPE	yes	yes	no	no	yes	no	no
DISTRIBUTE	yes	yes	no	no	yes	no	no
DTYPE	no	no	no	yes	no	no	no
DUNIT	no	no	no	yes	no	no	no
ENGINERESPONSETIMEOUT	yes	no	no	no	yes	no	no
ENVID	yes	no	no	no	yes	no	no
EXPIRATION	yes	no	no	no	yes	no	no
EXPIREDAYS	no	no	no	no	yes	no	no
FORMSIZE	no	no	no	yes	no	no	no
FROM	yes	yes	no	no	yes	no	no
GETJOBID	no	no	no	no	yes	no	no
GETSERVERINFO	no	no	no	no	yes	no	no
HELP	no	no	no	no	yes	no	no
ITEMTITLE	yes	yes	no	no	yes	no	no
JOBNAME	yes	no	no	no	yes	no	no
JOBRETRY							
JOBTYPE	yes	no	no	no	yes	no	no
JVMOPTIONS	no	yes	yes	yes	no	yes	no
KILLENGINE	no	no	no	no	yes	no	no
KILLJOBID	no	no	no	no	yes	no	no
LONGCHUNK	yes	yes	yes	no	yes	no	no
MIMETYPE	no	no	no	no	yes	no	no
MODE	yes	yes	no	no	yes	no	no
MODULE | REPORT	yes	yes	yes	no	yes	no	no
NAME	no	no	no	no	no	no	no
NONBLOCKSQL	yes	yes	yes	no	yes	no	no
NOTIFYFAILURE	yes	yes	no	no	yes	no	no
NOTIFYSUCCESS	yes	yes	no	no	yes	no	no
NUMBERFORMATMASK	yes	yes	no	no	yes	no	no
ONFAILURE	yes	yes	yes	no	yes	no	no
ONSUCCESS	yes	yes	yes	no	yes	no	no
ORIENTATION	yes	yes	no	no	yes	no	no
OUTPUTFOLDER*	yes	yes	no	no	yes	no	no

Table A–1 (Cont.) Keywords Usage Summary

Keywords	rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver	rwbridge
OUTPUTGRAPHFORMAT	yes	yes	yes	no	yes	no	no
OUTPUTIMAGEFORMAT	yes	yes	yes	no	yes	no	no
OUTPUTPAGE	yes	yes	no	no	yes	no	no
OVERWRITE	no	no	no	yes	no	no	no
P_AVAILABILITY	no	no	no	yes	no	no	no
P_DESCRIPTION	no	no	no	yes	no	no	no
P_FORMATS	no	no	no	yes	no	no	no
P_JDBCPDS	yes	yes	yes	no	yes	no	no
P_NAME	no	no	no	yes	no	no	no
P_OWNER	no	no	no	yes	no	no	no
P_PFORMTEMPLATE	no	no	no	yes	no	no	no
P_PRINTERS	no	no	no	yes	no	no	no
P_PRIVILEGE	no	no	no	yes	no	no	no
P_SERVERS	no	no	no	yes	no	no	no
P_TRIGGER	no	no	no	yes	no	no	no
P_TYPES	no	no	no	yes	no	no	no
PAGEGROUP	yes	yes	no	no	yes	no	no
PAGESIZE	yes	yes	yes	yes	yes	no	no
PAGESTREAM	yes	yes	no	no	yes	no	no
PARAMETER	yes	yes	yes	yes	yes	yes	yes
PARAMFORM	no	no	no	no	yes	no	no
PARSEQUERY	no	no	no	no	yes	no	no
PDFCOMP	yes	yes	no	no	yes	no	no
PDFEMBED	yes	yes	no	no	yes	no	no
PDFOWNER	yes	yes	yes	no	yes	no	no
PDFSECURITY	yes	yes	yes	no	yes	no	no
PDFUSER	yes	yes	yes	no	yes	no	no
PFACTION	yes	no	yes	no	no	no	no
PRINTJOB	no	no	yes	no	no	no	no
READONLY	yes	yes	yes	no	yes	no	no
RECURSIVE_LOAD	yes	yes	no	yes	yes	no	no
REPLYTO	yes	yes	no	no	yes	no	no
REPORT | MODULE	yes	yes	yes	no	yes	no	no
ROLE	yes	yes	no	no	yes	no	no
RUNDEBUG	yes	yes	yes	no	yes	no	no
SAVE_RDF	no	yes	yes	no	no	no	no
SCHEDULE	yes	no	no	no	yes	no	no
SERVER	yes	no	no	no	yes	yes	no
SHOWAUTH	no	no	no	no	yes	no	no
SHOWENV	no	no	no	no	yes	no	no
SHOWJOBID	no	no	no	no	yes	no	no
SHOWJOBS	no	no	no	no	yes	no	no

Table A–1 (Cont.) Keywords Usage Summary

Keywords	rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver	rwbridge
SHOWMAP	no	no	no	no	yes	no	no
SHOWMYJOBS	no	no	no	no	yes	no	no
SHUTDOWN	no	no	no	no	no	no	no
SITENAME*	yes	yes	no	no	yes	no	no
SOURCE	no	no	no	yes	no	no	no
SQLTRACE	yes	yes	yes	no	yes	no	no
SSOCONN	no	no	no	no	yes	no	no
STATUSFOLDER*	yes	yes	no	no	yes	no	no
STATUSFORMAT	no	no	no	no	yes	no	no
STATUSPAGE	yes	yes	no	no	yes	no	no
STYPE	no	no	no	yes	no	no	no
SUBJECT	yes	yes	no	no	yes	no	no
SUPPRESSLAYOUT	yes	yes	yes	no	yes	no	no
TOLERANCE	yes	no	no	no	yes	no	no
URLPARAMETER	yes	no	no	no	yes	no	no
USEJVM	yes	no	no	no	no	no	no
USERID	yes	yes	yes	yes	yes	no	no
USERSTYLES	yes	yes	yes	no	yes	no	no
VALIDATETAG	no	no	yes	no	no	no	no
WEBSERVER_DEBUG	no	no	yes	no	no	no	no
WEBSERVER_DOCROOT	no	no	yes	no	no	no	no
WEBSERVER_PORT	no	no	yes	no	no	no	no

A.4 Command-Line Keywords

For ease of navigation, the alphabetically ordered list of keywords is divided into the following sections:

- Command Line Keywords (ACCESSIBLE to DESTYPE)
- Command Line Keywords (DISTRIBUTE to ORIENTATION)
- Command Line Keywords (OUTPUTFOLDER to ROLE)
- Command Line Keywords (RUNDEBUG to WEBSERVER_PORT)

A.5 Command Line Keywords (ACCESSIBLE to DESTYPE)

This section describes each of the command line keywords that can be used in Oracle Reports.

A.5.1 ACCESSIBLE

Table A–2 indicates which components can use the `ACCESSIBLE` keyword.

Table A–2 Components That Use ACCESSIBLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description Use ACCESSIBLE to specify whether accessibility-related features offered through Oracle Reports are enabled (YES) or disabled (NO) for PDF output.

For detailed information about PDF in Oracle Reports, see Chapter 11, "Using PDF in Oracle Reports".

Syntax ACCESSIBLE={YES|NO}

Values

- YES Accessibility features are enabled for PDF output.
- NO Accessibility features are not enabled for PDF output.

Default NO

A.5.2 ARRAYSIZE

Table A–3 indicates which components can use the ARRAYSIZE keyword.

Table A–3 Components That Use ARRAYSIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description Use ARRAYSIZE to specify the size (in kilobytes) for use with Oracle's 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.5.3 AUTHID

Table A–4 indicates which components can use the AUTHID keyword.

Table A–4 Components That Use AUTHID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver	rwbridge
yes	yes	no	no	yes	yes	yes

Description

- Use AUTHID to specify the user name and 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
- With rwbridge: Use AUTHID to specify the user name and the password to authorize shutting down the Oracle Reports Bridge. You can set the identifier

element in the Oracle Reports Bridge configuration (using Oracle Enterprise Manager) to the administrator user name and password to secure the Oracle Reports Bridge. This ensures that only administrators can shut down the Oracle Reports Bridge.

Syntax AUTHID=*username*/*password*

Values

- *username*/*password* Any valid user name and password created in Oracle Portal. See your DBA to create new users accounts in Oracle Portal.
- With rwbridge:

  username/*password* The user name and password specified in the identifier element in the Oracle Reports Bridge configuration file (rwbridge_*bridgename*.conf).

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. It is recommended that you use Single Sign-On.
- In a Single Sign-On environment, when using Oracle Access Manager 11g (OAM) as the authentication server, if AUTHID is passed in the Reports URL, then OAM authentication page is displayed.

A.5.4 AUTOCOMMIT

Table A–5 indicates which components can use the AUTOCOMMIT keyword.

Table A–5 Components That Use AUTOCOMMIT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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 a URL.

A.5.5 BACKGROUND

Table A–6 indicates which components can use the BACKGROUND keyword.

Table A–6 Components That Use BACKGROUND

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	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, a warning is issued and the keyword is ignored.

A.5.6 BATCH

Table A–7 indicates which components can use the BATCH keyword.

Table A–7 Components That Use BATCH

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	no	yes

Description Use BATCH when you want the server to run in no-UI mode. No user interface is displayed 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 box displays while it is running.

With rwconverter, BATCH=YES suppresses all terminal input and output in order to convert reports and libraries without user intervention. With rwserver, BATCH turns the server dialog box 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 DESTYPE.
- If BATCH=YES, PARAMFORM=YES is invalid because it is not meaningful to have the Runtime Parameter Form appear in batch mode.

A.5.7 BCC

Table A–8 indicates which components can use the BCC keyword.

Table A–8 Components That Use BCC

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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

- Enclose each address in quotation marks. To specify more than one e-mail address, separate each address with a comma.
- Related keywords include 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.5.8 BLANKPAGES

Table A–9 indicates which components can use the BLANKPAGES keyword.

Table A–9 Components That Use BLANKPAGES

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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 Prints all blank pages.
- NO Does 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.5.9 BUFFERS

Table A–10 indicates which components can use the BUFFERS keyword.

Table A–10 Components That Use BUFFERS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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, for example, 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.5.10 CACHELOB

Table A–11 indicates which components can use the CACHELOB keyword.

Table A–11 Components That Use CACHELOB

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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, *rwserver*.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.5.11 CC

Table A–12 indicates which components can use the CC keyword.

Table A–12 Components That Use CC

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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

- Enclose each address in quotation marks. To specify more than one e-mail address, separate each address with a comma.
- Related keywords include BCC, FROM, REPLYTO, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.5.12 CELLWRAPPER

Table A–13 indicates which components can use the CELLWRAPPER keyword.

Table A–13 Components That Use CELLWRAPPER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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.5.13 CMDFILE

Table A–17 indicates which components can use the CMDFILE keyword.

Table A–17 Components That Use CMDFILE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	yes	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 key map file (see Section 18.14, "Using a Key Map 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, use the CMDKEY keyword to refer to a key in the cgicmd.dat file instead 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.5.14 CMDKEY

Table A–18 indicates which components can use the CMDKEY keyword.

Table A–18 Components That Use CMDKEY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use CMDKEY to call a key-identified command line in the cgicmd.dat key map file. See Section 18.14, "Using a Key Map File". For example:

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.

Example

```
http://your_webserver/reports/rwservlet?cmdkey=key& ...
```

A.5.15 COLLATE

Table A–19 indicates which components can use the COLLATE keyword.

Table A–19 Components That Use COLLATE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use COLLATE to control the collating behavior when a report is output to a printer.

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

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

A.5.16 COMPILE_ALL

Table A–20 indicates which components can use the COMPILE_ALL keyword.

Table A–20 Components That Use COMPILE_ALL

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	no	no

Description Use COMPILE_ALL to forcibly compile all program units in the report being converted by rwconverter, except in the following cases:

- If the destination type (DTYPE) is REXFILE, XMLFILE, or JSPFILE, then rwconverter will not compile any program units. When REX, XML, or JSP report definitions are opened in Oracle Reports Builder or Reports Server, they are automatically compiled by Oracle Reports.

Syntax COMPILE_ALL={YES|NO}

Values

- YES Compiles all program units, except in cases where DTYPE is REXFILE, XMLFILE, or JSPFILE.
- NO Compiles only uncompiled program units.

Default NO

Usage Notes By default, rwconverter compiles all uncompiled program units during the conversion operation. When COMPILE_ALL=YES, then rwconverter forcibly compiles all the program units (including those already compiled) in the report. This may be useful when moving a report to a different client machine, to ensure everything is recompiled and avoid potential incompatibilities.

A.5.17 CONTAINSHTMLTAGS

Table A–23 indicates which components can use the CONTAINSHTMLTAGS keyword.

Table A–21 Components That Use CONTAINSHTMLTAGS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description You can use a defined set of HTML formatting tags to format text style (bold, italics, underline, and strikethrough) and text attributes (font name, font color, and font size), and generate formatted text objects in all bitmap output formats supported by Oracle Reports when the objects' Contains HTML Tags property is set to Yes.

Use CONTAINSHTMLTAGS to specify whether Oracle Reports should interpret the HTML formatting tags for all the supported output formats.

Syntax CONTAINSHTMLTAGS=YES|NO

Values

- YES Oracle Reports interprets the HTML formatting tags for all objects whose Contains HTML Tags property is set to Yes.
- NO Oracle Reports does not interpret the HTML tags for the report, regardless of the object's Contains HTML Tags property setting. For HTML and HTMLCSS ouput, the browser will interpret the HTML formatting tags; for other output formats, the HTML tags themselves will appear as is in the report output.

Default YES

Usage Notes

- The supported output formats are: PDF, RTF, HTML, HTMLCSS, SPREADSHEET, and PostScript.

- Oracle Reports' interpretation of HTML tags may be different from the browser's interpretation. As a result, a report designed with HTML tags in releases prior to Oracle Reports 10g Release 2 (10.1.2) may generate different HTML or HTMLCSS output than later releases, where Oracle Reports interprets HTML formatting tags. If you do not wish for Oracle Reports to interpret HTML formatting tags, and instead retain the behavior of prior releases, set the REPORTS_CONTAINSHTMLTAGS environment variable to NO.
- If you set the REPORTS_CONTAINSHTMLTAGS environment variable to NO, you can still specify CONTAINSHTMLTAGS=YES on the command line for selected reports to have Oracle Reports interpret the HTML formatting tags for all the supported output formats. In other words, the value specified by this command line keyword overrides the REPORTS_CONTAINSHTMLTAGS environment variable.

A.5.18 CONTAINSOLE

Note: OLE support is obsolete in Oracle Reports (OLE is a client/server feature that is not applicable in a Web-based environment). Instead, use mime types with associated plug-ins and hyperlinks.

Table A–23 indicates which components can use the CONTAINSOLE keyword.

Table A–22 Components That Use CONTAINSOLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description For backward compatibility, use CONTAINSOLE to specify whether the program units or attached libraries for the report contain any OLE (Object Linking and Embedding) calls. If CONTAINSOLE=YES, the OLE system is initialized at the start of report execution and terminated at the end of that report execution.

Syntax CONTAINSOLE=YES|NO

Values

- YES The report includes OLE calls in program units or attached libraries.
- NO The report does not contain any OLE calls in program units or attached libraries.

Default NO

A.5.19 CONTENTAREA

Table A–23 indicates which components can use the CONTENTAREA keyword.

Table A–23 Components That Use CONTENTAREA

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use CONTENTAREA to specify the Oracle9*i*AS Portal Release 1 content area to which report output should be pushed. This keyword is maintained for backward

compatibility with Oracle9*i*AS Portal Release 1; for backward compatibility with Oracle WebDB Release 2.2, see SITENAME. Beginning with Oracle Portal 10*g* Release 1 (9.0.4), use PAGEGROUP.

Syntax CONTENTAREA=*name*

Values

name The name (*internal name*) of any valid Oracle9*i*AS Portal Release 1 content area.

Default None

Usage Notes

- Use of this keyword is *required* to push Oracle Reports output to Oracle9*i*AS 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 Oracle9*i*AS Portal Release 1 component instance.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.5.20 COPIES

Table A–24 indicates which components can use the COPIES keyword.

Table A–24 Components That Use COPIES

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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, for example, a comma or a decimal point).

Default Taken from the Initial Value property of the COPIES parameter (the Initial Value was defined in Oracle 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 1.

A.5.21 CUSTOMIZE

Table A–25 indicates which components can use the CUSTOMIZE keyword.

Table A–25 Components That Use CUSTOMIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	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 of the engine element).

> **Note:** For more information on customizing reports at runtime with XML customization files, see Chapter 22, "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.
- 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 include CUSTOMIZE as the last keyword when specified in a URL; for example:

  ```
  ...REPORT=emp.rdf CUSTOMIZE=c:\myreports\emp.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 preceding example, you could switch the positions of the REPORT and CUSTOMIZE parameters in your URL.

A.5.22 DATEFORMATMASK

Table A–26 indicates which components can use the DATEFORMATMASK keyword.

Table A–26 Components That Use DATEFORMATMASK

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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 *Oracle Reports online Help* topic, "Date and Time Format Mask Syntax."

- DATEFORMATMASK can be used when running JSP-based Web reports from the command line.

A.5.23 DBPROXYCONN

Table A–26 indicates which components can use the DBPROXYCONN keyword.

Table A–27 Components That Use DBPROXYCONN

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use DBPROXYCONN to specify The key to be used for obtaining proxy user name, password, and database information from Oracle Internet Directory. This key is created in Oracle Internet Directory when specifying default Resource Access Information.

Syntax dbproxyconn=key

Values

KEY The resource name configured in Oracle Internet Directory in the user RAD or the default RAD.

Default None

Usage Notes

You can add the dbproxy connection keys in the server configuration files.

A.5.24 DELAUTH

Table A–28 indicates which components can use the DELAUTH keyword.

Table A–28 Components That Use DELAUTH

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use DELAUTH to delete rwservlet user ID cookies.

Syntax `http://your_webserver/reports/rwservlet/delauth[?]`
`[server=server_name][&authid=username/password]`

Values See Syntax

Default None

Usage Notes

- This keyword is a command that does not require a value; that is, commands are entered by themselves without a corresponding value.
- Related keywords are SERVER and AUTHID.

A.5.25 DELIMITED_HDR

Table A–29 indicates which components can use the DELIMITED_HDR keyword.

Table A–29 Components That Use DELIMITED_HDR

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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 Leave boilerplate text as is in the delimited output file.
- NO Turn off all boilerplate text 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.5.26 DELIMITER

Table A–30 indicates which components can use the DELIMITER keyword.

Table A–30 Components That Use DELIMITER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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–31 Valid Values - General

Values	Description
,	A comma separates each cell
.	A period separates each cell

Any of these reserved values:

Table A–32 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–33 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.5.27 DESFORMAT

Table A–34 indicates which components can use the DESFORMAT keyword.

Table A–34 Components That Use DESFORMAT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use DESFORMAT to specify either the output format for the report, or the printer definition to use when formatting the report when DESTYPE=FILE and DESNAME=*filename*.

Syntax DESFORMAT=*desformat*

Values Any valid destination format not to exceed 1K in length. Examples of valid values for this keyword are listed and described in Table A–35.

Table A–35 Valid Values for DESFORMAT

Value	Description
DFLT	The report output is sent to a file that uses the default printer driver to format the report (for example, a PostScript driver generates PostScript output format).

Table A–35 (Cont.) 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 (through the DELIMITER keyword), the default delimiter is a tab. See Usage Notes.
DELIMITEDDATA	Provides similar functionality as DELIMITED, and is used when you have problems running large volume reports with DESFORMAT=DELIMITED. See Usage Notes.
HTML	The report output is sent to a file that is in HTML format. See Usage Notes.
HTMLCSS	The report output is sent to a file that includes style sheet extensions. See Usage Notes.
PDF	The 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.
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. A value of DFLT means the default printer driver is used. If MODE=CHARACTER, this is the name of a 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 definitions.
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. See Usage Notes.
SPREADSHEET	(Command line only) The report output is sent to an HTML file that can be directly opened with Microsoft Excel 2000. You can generate spreadsheet output from the paper layout of reports saved in any format (.rdf, .jsp, .xml). See Usage Notes.
ENHANCEDSPREADSHEET	(Command line only) In Oracle Reports, the report output is sent to an HTML file that is compatible with spreadsheet applications, such as Microsoft Excel, and also provides enhancements to the SPREADSHEET output format to enable you to burst and distribute reports to spreadsheet format, as well as generate large data sets (up to 75,000 rows) to spreadsheets. You can generate spreadsheet output from the paper layout of reports saved in any format (.rdf, .jsp, .xml). See Usage Notes.
XML	The 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 system parameter (defined in Oracle Reports Builder at design time). When you run a report through Oracle 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, then Oracle Reports Builder defaults to PostScript output format.

Usage Notes

- The value(s) 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, which is specified on the command line with the DESTINATION keyword to distribute the report. In this case, Oracle Reports displays an error:

  ```
  REP-34305: Invalid keyword setting for the destid='DEST1'
  ```

  Note: DST files are supported for backward compatibility; the preferred and recommended method of distributing reports is with the Distribution dialog box in Reports Builder, or using XML as described in Chapter 20, "Creating Advanced Distributions".

  The DELIMITED functionality also honors the DELIMITER, CELLWRAPPER, NUMBERFORMATMASK, and DATEFORMATMASK command line keywords.
- When DESFORMAT=DELIMITEDDATA, the DelimitedData driver runs off the report data model and operates in much the same way as the XML driver. Since the driver runs off the data model, any formatting 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 keywords just as DELIMITED does.

  For more information on delimited output, see "About delimited output" in the *Oracle Reports online Help* (and also in the "Advanced Concepts" chapter in the *Oracle Reports User's Guide to Building Reports* manual).
- When DESFORMAT=SPREADSHEET, the report output preserves the rich layout formatting such as colors, fonts, conditional formatting, graphs, and images. For detailed information about how different report objects are generated in a report run to DESFORMAT=SPREADSHEET, see "About Spreadsheet Output" in the *Oracle Reports online Help* (and also in the "Advanced Concepts" chapter in the *Oracle Reports User's Guide to Building Reports* manual).
- 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.5.28 DESNAME

Table A–36 indicates which components can use the DESNAME keyword.

Table A–36 Components That Use DESNAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use DESNAME to specify the name of the cache, file, printer, WebDAV server, 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 WebDAV server, 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 Oracle 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:

  ```
  tsmith@companya.com,gjones@companyb.com,mroberts@companyc.com
  ```

- In some cases, this keyword may be overridden by your operating system.

Examples

Example 1: Sending report output to a file

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf destype=file
desname=c:\mydir\test.pdf

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=pdf+destype=file+desname=c:\mydir\test.pdf

rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb desformat=pdf
destype=file desname=c:\mydir\test.
```

Example 2: Sending report output to a printer

```
rwrun report=test.rdf userid=scott/tiger@mydb destype=printer
desname=myprinter

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+destype=printer+desname=myprinter

rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb destype=printer
```

```
desname=myprinter
```

Example 3: Sending report output to 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"

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@comp.com"+cc="emp3@comp.com"+bcc="mgr@comp.com"+
replyto="me@comp.com"+from="me@comp.com"

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 4: Sending report output to WebDAV (any WebDAV server or Oracle Portal WebDAV)

> **Note:** Currently there is no support for FTP and WebDAV destinations from the Reports Builder environment. However, they are supported from the Reports Runtime and the Reports Server environments.

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=htmlcss destype=webdav
desname="http://myusername:mypassword@mywebdavserv.com/mydir/test.html"

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=htmlcss+destype=webdav+
desname="http://myusername:mypassword@mywebdavserv.com/mydir/test.html"

rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydbdesformat=htmlcss
destype=webdav
desname="http://myusername:mypassword@mywebdavserv.com/mydir/test.htm"
```

A.5.29 DEST

Table A–37 indicates which components can use the DEST keyword.

Table A–37 Components That Use DEST

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 (for example, (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 TDFFILE, then the DEST default name is source.tdf.
- If DTYPE is XMLFILE, then the DEST default name is source.xml.
- If DTYPE is JSPFILE, then the DEST default name is source.jsp.
- If DTYPE is REGISTER, then the DEST default name is the name of the SQL*Plus script output file (for example, 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.5.30 DESTINATION

Table A–38 indicates which components can use the DESTINATION keyword.

Table A–38 Components That Use DESTINATION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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.
- 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 defining the distribution for a report because your URL might end with the DESTINATION keyword. For example:

  ```
  ...DISTRIBUTE=yes DESTINATION=c:\oracle\reports\dist\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 preceding example, you could switch the positions of the DISTRIBUTE and DESTINATION parameters in your URL.

> **Note:** For more information on creating advanced distributions, see Chapter 20, "Creating Advanced Distributions".

A.5.31 DESTYPE

Table A–39 indicates which components can use the DESTYPE keyword.

Table A–39 Components That Use DESTYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use DESTYPE to specify the type of device that will receive the report output for paper-based reports. If you have created your own pluggable destination through the Oracle Reports Destination API, this is how the destination you created gets called.

Syntax DESTYPE={CACHE|LOCALFILE|FILE|PRINTER|MAIL|ORACLEPORTAL|FTP|WEBDAV|*name_of_pluggable_destination*}

Values Table A–40 lists and describes the valid values for the DESTYPE keyword.

Table A–40 Valid Values for DESTYPE

Value	Description
CACHE	Valid only for rwclient and rwservlet. Sends the output directly to the Web browser (cache).
LOCALFILE	Valid only for rwclient and rwservlet. Sends the output to a file on the client machine, synchronously or asynchronously. When used with rwclient, DESTYPE=LOCALFILE saves the output to the client machine using the file name specified by DESNAME. When used with rwservlet, DESTYPE=LOCALFILE sets the mimetype to application/octet-stream to force the browser to display the Save dialog box. If for some reason this does not work, you can instead specify DESTYPE=CACHE and add MIMETYPE=REPORTS/LOCAL (or any non-registered mimetype) to force the browser to display the Save dialog box.

Table A–40 (Cont.) Valid Values for DESTYPE

Value	Description
FILE	Sends the output to the file on the server named in DESNAME.
PRINTER	Sends the output to the printer on the server named in DESNAME. You must have a printer that Oracle Reports Services can recognize installed and running. See Usage Notes, below.
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 *rwserver*.conf must include the outgoing mail server name. This applies in both Windows and UNIX environments. Refer to Section 7.2.1.2, "pluginParam".
ORACLEPORTAL	Valid only for rwclient and rwservlet. Sends the output to Oracle Portal. Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SCHEDULE, SITENAME*,STATUSFOLDER*,STATUSPAGE. * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2. See Usage Notes, below.
FTP	Sends the output to the specified FTP server. See Usage Notes, below.
WEBDAV	Sends the output to the specified WebDAV server so that the report can be published directly. See Usage Notes, below.
name_of_pluggable_destination	If you have created your own pluggable destination through the Oracle Reports Destination API, this is what you use to call the destination you created.

Default Taken from the Initial Value property of the DESTYPE system parameter (defined in Oracle Reports Builder at design time).

Usage Notes

- DESTYPE values of SCREEN and PREVIEW are no longer valid because the Reports Runtime (rwrun) user interface is obsolete. In Oracle Reports Builder, you can still set the DESTYPE system parameter to SCREEN to format a report to display screen fonts in the Previewer in the Oracle 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 Oracle Reports Services. To revert to the old behavior of including the hardware margin, set the REPORTS_ADD_HWMARGIN environment variable to YES.
- DESTYPE=ORACLEPORTAL: Before you push Oracle Reports output to Oracle Portal, 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 must edit the Reports Server configuration file (`rwserver.conf`) 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%"
             encrypted="no"/-->
</destination>
```

Note: By default, the `portalUserid` is commented out. Reports Server will determine the connection string and push the report to Oracle Portal. You must uncomment this only if you are using a different Oracle Portal instance.

---

2. Substitute the values in the `portalUserid` property with your Oracle Portal connection information if you do not want to push Oracle Reports output to the default Oracle Portal 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&auth
id=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 pluggable destinations. You must 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 this example, the `DESTYPE` is `FTP` and the `DESNAME` value is a complete FTP URL location along with the report name `myreport.pdf`.

To specify proxy information to send and receive information through a firewall, see Section 7.7, "Entering Proxy Information".

Note: The proxy server specified for the FTP destination must support the SOCKS protocol. This check is performed during initialization. If the proxy server does not support the SOCKS protocol, then the server raises the following error:

```
REP-62352: FTP Proxy Server specified is not responding
```

- DESTYPE=WEBDAV: Running the request is very similar to any other out-of-the-box plyuggable destinations. You must 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://user:pwd@webdavserver/myreport.pdf&desformat=pdf**

  In this example, the DESTYPE is WEBDAV and the DESNAME value is a complete WebDAV URL location along with the report name myreport.pdf.

  To specify proxy information to send and receive information through a firewall, see Section 7.7, "Entering Proxy Information".

Examples

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

rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb desformat=pdf
destype=cache
```

Example 2: Sending report output to a file

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=pdf destype=file
desname=c:\mydir\test.pdf

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=pdf+destype=file+desname=c:\mydir\test.pdf

rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb desformat=pdf
destype=file desname=c:\mydir\test.pdf
```

Example 3: Sending report output to a printer

```
rwrun report=test.rdf userid=scott/tiger@mydb destype=printer
desname=myprinter

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+destype=printer+desname=myprinter

rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb destype=printer
desname=myprinter
```

Example 4: Sending report output to 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"

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@comp.com"+cc="emp3@comp.com"+bcc="mgr@comp.com"+
replyto="me@comp.com"+from="me@comp.com"

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 report output to WebDAV (any WebDAV server or Oracle Portal WebDAV)

```
rwrun report=test.rdf userid=scott/tiger@mydb desformat=htmlcss destype=webdav
desname="http://myusername:mypassword@mywebdavserv.com/mydir/test.html"

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+desformat=htmlcss+destype=webdav+
desname="http://myusername:mypassword@mywebdavserv.com/mydir/test.html"

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 report output to Oracle Portal

```
rwrun report=test.rdf userid=scott/tiger@mydb destype=oracleportal desformat=PDF
pagegroup=mypagegrp outputpage=reports_output itemtitle=pushtoportal
statuspage=result

http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.rdf+
userid=scott/tiger@mydb+destype=oracleportal+desformat=PDF+pagegroup=mypagegrp+
outputpage=reports_output+itemtitle=pushtoportal+statuspage=result

rwclient server=myrepserv report=test.rdf userid=scott/tiger@mydb
destype=oracleportal desformat=PDF pagegroup=mypagegrp outputpage=reports_output
itemtitle=pushtoportal statuspage=result
```

A.6 Command Line Keywords (DISTRIBUTE to ORIENTATION)

This section provides a brief description of the Oracle Reports components and the keywords that each component can use.

A.6.1 DISTRIBUTE

Table A–41 indicates which components can use the `DISTRIBUTE` keyword.

Table A–41 Components That Use DISTRIBUTE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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 the Distribution dialog box in Oracle 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 on creating advanced distributions, see Chapter 20, "Creating Advanced Distributions".

A.6.2 DTYPE

Table A–42 indicates which components can use the DTYPE keyword.

Table A–42 Components That Use DTYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 Oracle Portal with the RWWWVREG.REGISTER_REPORT function. Each load function is populated with the necessary information to register the report in Oracle Portal. By running the resulting script file in SQL*Plus against the Oracle Fusion Middleware DB Provider, you can batch register multiple reports in Oracle Portal. See Appendix C, "Batch Registering Reports in Oracle 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, ensure that you compile the source report's PL/SQL using **Program > Compile** in Oracle 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.6.3 DUNIT

Table A–43 indicates which components can use the DUNIT keyword.

Table A–43 Components That Use DUNIT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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.6.4 ENGINERESPONSETIMEOUT

Table A–44 indicates which command can use the ENGINERESPONSETIMEOUT keyword.

Table A–44 Components That Use ENGINERESPONSETIMEOUT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	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 (for example, 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 Section 7.2.1.8, "engine" for information about the engine element.

Syntax ENGINERESPONSETIMEOUT=*number*

Values

number A number of minutes (for example, 5).

Default None

A.6.5 ENVID

Table A–44 indicates which command can use the ENVID keyword.

Table A–45 Components That Use ENVID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	yes	no

Description Use ENVID to specify the environment required for the current job request. This keyword allows for dynamic environment switching, as described in Section 7.2.2, "Dynamic Environment Switching".

Syntax ENVID=*id*

Values

id An identifier that corresponds to an environment element id in the configuration file. The matching environment element defines environment variables that will be used for the current job request. For examples, see Section 7.2.2, "Dynamic Environment Switching".

Default None

A.6.6 EXPIRATION

Table A–46 indicates which command can use the EXPIRATION keyword.

Table A–46 Components That Use EXPIRATION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	yes	no

Description Use EXPIRATION to define how long report output can exist in cache before it is deleted.

See Section 18.13, "Reusing Report Output from Cache" for more information on duplicate job detection. See Section 24.3, "Tuning Reports Server Configuration" and Section 24.7, "Running the Report" for tuning considerations in relation to maxQueueSize and cacheSize values.

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 time zone 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.6.7 EXPIREDAYS

Table A–47 indicates which components can use the EXPIREDAYS keyword.

Table A–47 Components That Use EXPIREDAYS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use EXPIREDAYS to specify the number of days after which the Oracle Reports output pushed to Oracle Portal 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 Oracle Portal.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*,

STATUSFOLDER*, STATUSPAGE.

* maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.6.8 FORMSIZE

Table A–48 indicates which components can use the FORMSIZE keyword.

Table A–48 Components That Use FORMSIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 (for example, 8.5 x 11).
- For more information on the Runtime Parameter Form, see the PARAMFORM keyword.

A.6.9 FROM

Table A–49 indicates which components can use the FROM keyword.

Table A–49 Components That Use FROM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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, REPLYTO, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.6.10 GETJOBID

Table A–50 indicates which components can use the GETJOBID keyword.

Table A–50 Components That Use GETJOBID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use GETJOBID to retrieve the result output of the Reports Server job with job ID *n*.

Syntax `http://your_webserver/reports/rwservlet/getjobid n[?][server=server_name][&authid=username/password]`

Values See Syntax

Default None

Usage Notes

- This keyword is a command that does not require a value; that is, 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.6.11 GETSERVERINFO

Table A–51 indicates which components can use the GETSERVERINFO keyword.

Table A–51 Components That Use GETSERVERINFO

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use GETSERVERINFO to display Reports Server information. You can choose the format (HTML or XML) in which the information is returned through statusformat.

Syntax `http://your_webserver/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; that is, commands are entered by themselves without a corresponding value.
- Related keywords are SERVER and AUTHID.

A.6.12 HELP

Table A–52 indicates which components can use the HELP keyword.

Table A–52 Components That Use HELP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use HELP to display a help topic that lists the keywords you can use with the rwservlet command.

Syntax http://*yourwebserver*/reports/rwservlet/help

or

http://*your_webserver*/reports/rwservlet/help?*command=keyword*

Values See Syntax

Default None

Example

http://*your_webserver*/reports/rwservlet/help?*command=keyword*

A.6.12.1 SQL Injection problem (Reports Help)

SQL injection attack is the insertion of an SQL statement via input data from user to the application. It allows attackers to create, read, update, or delete any arbitrary data available in the database, restricted only by database user privileges.

A user may execute SQL statements against the help database to retrieve sensitive information.A user can do this by adding a SQL statement in the command parameter of the help page for the reports server. A malicious user will be able to extend this attack to even more dangerous SQL statements.This may cause damage to the database data or operating system depending on privileges of the database user used by the application.

URL: *http://xxx.xxx.xxx.xxx:9002/reports/rwservlet/helps?commond=*

Note: Follow this steps to reproduce SQL injection vulnerability:

1. Enter the following URL in a browser: *http://xxx.xxx.xxx.xxx:9002/reports/rwservlet/help?command=delauth'*. Oracle Reports error page is displayed due to the trailing single quote.
2. Enter the following URL in a browser: *http://xxx.xxx.xxx.xxx:9002/reports/rwservlet/help?command=delauth"*. When two single quotes are used, a blank page is displayed. This change in response indicates that the user supplied value was incorporated directly in a database query and that a SQL injection vulnerability is present.

To resolve this vulnerability, prepared statement objects to avoid the risk posed by this type of attack. Also, implement positive validation or white-list filtering to accept only known "good" input values and filter out the rest.

A.6.13 ITEMTITLE

Table A–53 indicates which components can use the ITEMTITLE keyword.

Table A–53 Components That Use ITEMTITLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use ITEMTITLE to specify the display name Oracle Portal should use for Oracle Reports output. The name will display in Oracle Portal 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 key map file. See Section 18.14, "Using a Key Map File".

Default The report file name

Usage Notes

- Use of this keyword is *optional* to push Oracle Reports output to Oracle Portal.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.6.14 JOBNAME

Table A–54 indicates which components can use the JOBNAME keyword.

Table A–54 Components That Use JOBNAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	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

string Any job name.

Default None

Usage Notes JOBNAME can be used when running JSP-based Web reports from the command line.

A.6.15 JOBRETRY

Table A–55 indicates which components can use the JOBRETRY keyword.

Table A–55 Components That Use JOBRETRY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	yes	no

Description Use JOBRETRY to specify the maximum number of times a job should be retried after failing. When specified, this keyword takes precedence over the server configuration file entry for the retry attribute of the job element.

Syntax JOBRETRY=*retries*

Values

retries An integer that specifies the number of times to retry a job that fails with unexpected errors, after the initial run (total attempts = initial run + JOBRETRY value).

Default 0

Usage Notes Jobs explicitly canceled are not be retried. All jobs that fail with unexpected errors (either engine crash or normal errors) are retried.

If an invalid value is specified for JOBRETRY, Oracle Reports generates runtime exception REP-50003.

A.6.16 JOBTYPE

Table A–56 indicates which components can use the JOBTYPE keyword.

Table A–56 Components That Use JOBTYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	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 Reports Server has an engine to process it.

Syntax JOBTYPE=*jobtype*

Values

jobtype A job for which Reports Server has an engine. For example: report (for rwEng engine) or rwurl (for rwURLEng engine).

Default report

Usage Notes The database authentication functionality provided in Oracle Reports is available only when JOBTYPE=report. This is the job type of the default engine (rwEng) provided with OracleAS Reports Services. The database authentication functionality is not available when JOBTYPE specifies a different value (for example, for a custom engine that you develop yourself). This is because a custom engine may require a different format for the connect string, while the Oracle Reports database authentication functionality limits the connect string to the Oracle Reports format user/password@dbname used for the default engine.

A.6.17 JVMOPTIONS

Table A–57 indicates which components can use the JVMOPTIONS keyword.

Table A–57 Components That Use JVMOPTIONS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	yes	yes	yes	no	yes

Description Use JVMOPTIONS to set options for the Java Virtual Machine (JVM).

Syntax

```
JVMOPTIONS={options in the Reports Runtime, Reports Builder, Reports Converter, or Reports Server's JVM}
```

Usage Notes

- The default value -Xmx256M specifies the JVM heap size of 256 MB to avoid the Out Of Memory error when running reports with large graphs or running big reports.
- When the Reports Engine starts up, it checks for JVM options specified in the *rwserver*.conf file in the jvmoptions attribute of the engine element. See Section 7.2.1.8, "engine". If specified, the JVM options set in *rwserver*.conf override the value of the REPORTS_JVM_OPTIONS environment variable. If not specified in *rwserver*.conf, Oracle Reports uses the JVM options specified by the REPORTS_JVM_OPTIONS environment variable. See Section B.1.55, "REPORTS_JVM_OPTIONS".
- When running reports with Reports Server, JVM options cannot be set using the REPORTS_JVM_OPTIONS environment variable. For Reports Server, set JVM options on the command line using the JVMOPTIONS command line keyword.
- When running reports with Reports Builder, Reports Runtime, and Reports Converter, JVM options specified on the command line with the JVMOPTIONS command line keyword override JVM options specified by the REPORTS_JVM_OPTIONS environment variable.

Examples

You could use the following command line to start the Reports Server (rwserver) with a 512MB heap space:

```
rwserver server=servername jvmoptions=-Xmx512M
```

You could also use the following command line to start Oracle Reports Builder (rwbuilder) with a 512MB heap space:

```
rwbuilder jvmoptions=-Xmx512M
```

If multiple options are passed, they must be enclosed in quotes:

```
rwserver server=servername jvmoptions="-Xmx256M -Xms128M"
```

A.6.18 KILLENGINE

Table A–58 indicates which components can use the KILLENGINE keyword.

Table A–58 Components That Use KILLENGINE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use KILLENGINE to stop a Reports Server engine with the specified engine ID and engine type. For a secured Reports Server, only users with Administrator privileges can use this keyword. For an unsecured Reports Server, the user ID and password values for the AUTHID keyword must match the user ID and password specified by the identifier tag in the *server*.conf configuration file.

Syntax http://*your_webserver*/reports/rwservlet/killengine[?][server=*server_name*][&authid=*username*/*password*] [&type=*engine_type*]

Values See Syntax

Default None

Usage Notes

- The engine must currently exist in the Reports Server.
- Use GETSERVERINFO to see the current engines existing in the server.
- Related keywords are GETSERVERINFO, SERVER, and AUTHID.

Example

To kill an engine rwEng-0

http://*yourwebserver*/reports/rwservlet/killengine0?server=myserver&authid=mydb/*password*&type=rwEng

A.6.19 KILLJOBID

Table A–59 indicates which components can use the KILLJOBID keyword.

Table A–59 Components That Use KILLJOBID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use KILLJOBID to kill a Reports Server job with the specified job ID *n*.

Syntax http://*your_webserver*/reports/rwservlet/killjobid*n*[?] [server=*server_name*][&authid=*username*/*password*] [&statusformat={html|xml|xmldtd}]

Values See Syntax

Default None

Usage Notes

- 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.

A.6.20 LONGCHUNK

Table A–60 indicates which components can use the LONGCHUNK keyword.

Table A–60 Components That Use LONGCHUNK

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description LONGCHUNK is the size (in kilobytes) of the increments in which Oracle 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, for example, 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.6.21 MIMETYPE

Table A–61 indicates which components can use the MIMETYPE keyword.

Table A–61 Components That Use MIMETYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	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.

Syntax MIMETYPE=*string*

Values

string A valid MIME type specification.

Default None

Usage Notes Oracle 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.

Example

```
MIMETYPE=application/vnd.ms-excel
```

A.6.22 MODE

Table A–62 indicates which components can use the MODE keyword.

Table A–62 Components That Use MODE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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 Oracle Reports Builder.

Syntax MODE={BITMAP|CHARACTER|DEFAULT}

Values

- BITMAP Run the report in bitmap mode.
- DEFAULT Run the report in the mode of the current component being used.
- CHARACTER On Windows - the Oracle Reports Builder ASCII driver will be used to produce editable ASCII output.

Default DEFAULT

A.6.23 MODULE|REPORT

Table A–63 indicates which components can use the MODULE|REPORT keyword.

Table A–63 Components That Use MODULE|REPORT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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, Oracle 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 (for example, display it in the Paper Design View), it must be a complete report definition (that is, contain its own data model and layout definition). You cannot run a partial report definition.
- An XML report definition *must* have the .xml file extension when specified with the MODULE|REPORT keyword.

- If you do not enter a file extension, Oracle Reports Builder 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.6.24 NAME

The NAME keyword is used only by the rwbridge component.

Description Use NAME to specify the name of the Oracle Reports Bridge. The Oracle Reports Bridge component (rwbridge) searches for the Oracle Reports Bridge configuration file, repbrg_*bridgename*.conf, in *ORACLE_HOME*/reports/conf. If not found, a new configuration file is created in *ORACLE_HOME*/reports/conf.

Syntax NAME=*bridgename*

Values

bridgename Any alphanumeric string.

Default None

A.6.25 NONBLOCKSQL

Table A–64 indicates which components can use the NONBLOCKSQL keyword.

Table A–64 Components That Use NONBLOCKSQL

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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.6.26 NOTIFYFAILURE

Table A–65 indicates which components can use the NOTIFYFAILURE keyword.

Table A–65 Components That Use NOTIFYFAILURE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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. See the notification discussion in Configuring Oracle 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 Fusion Middleware. 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.6.27 NOTIFYSUCCESS

Table A–66 indicates which components can use the NOTIFYSUCCESS keyword.

Table A–66 Components That Use NOTIFYSUCCESS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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 in the configuration file, or when you select the Enable Email Notification element on the Reports Application Advanced Configuration page in Oracle Enterprise Manager. See the notification discussion in Chapter 7, "Configuring Oracle 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 Fusion Middleware. 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.6.28 NUMBERFORMATMASK

Table A–67 indicates which components can use the NUMBERFORMATMASK keyword.

Table A–67 Components That Use NUMBERFORMATMASK

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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 *Oracle Reports online Help* topic, "Number Format Mask Syntax."

- NUMBERFORMATMASK can be used when running JSP-based Web reports from the command line.

A.6.29 ONFAILURE

Table A–68 indicates which components can use the ONFAILURE keyword.

Table A–68 Components That Use ONFAILURE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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, Oracle 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. See the READONLY command.
- ONFAILURE can be used when running JSP-based Web reports from the command line.

A.6.30 ONSUCCESS

Table A–69 indicates which components can use the ONSUCCESS keyword.

Table A–69 Components That Use ONSUCCESS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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, Oracle 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. See the READONLY command.
- ONSUCCESS can be used when running JSP-based Web reports from the command line.

A.6.31 ORIENTATION

Table A–70 indicates which components can use the ORIENTATION keyword.

Table A–70 Components That Use ORIENTATION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description ORIENTATION specifies 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

- The ORIENTATION command line keyword is effective only when DESTYPE=PRINTER or DESFORMAT=RTF. The orientation of RTF report output is based on the following order of precedence:
  1. ORIENTATION command line keyword.
  2. ORIENTATION system parameter.
  3. Orientation property for the pertinent report section (Header, Main, or Trailer): Portrait or Landscape.
  4. (If the Orientation property is set to Default) Width and Height properties for the pertinent report section:
     - If Width is greater than Height (for example, 11 x 8.5), then orientation is landscape.
     - If Height is greater than Width (for example, 8.5 x 11), then orientation is portrait.
- For PDF report output, any values specified for the ORIENTATION command line keyword, ORIENTATION system parameter, and Orientation property are ignored. Orientation of the PDF output is based solely on the values of the Width and Height properties, as above.
- If ORIENTATION=LANDSCAPE for a character-mode report, then you must ensure that your printer definition file contains a landscape clause.
- This keyword is not supported when output to a PCL printer on UNIX.

A.7 Command Line Keywords (OUTPUTFOLDER to ROLE)

This section provides a brief description of the Oracle Reports components and the keywords that each component can use.

A.7.1 OUTPUTFOLDER

Table A–71 indicates which components can use the OUTPUTFOLDER keyword.

Table A–71 Components That Use OUTPUTFOLDER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use OUTPUTFOLDER to specify the name of the Oracle WebDB Release 2.2 or Oracle9*i*AS Portal Release 1 folder to which to report output should be pushed. This keyword is maintained for backward compatibility with Oracle WebDB Release 2.2 and Oracle9*i*AS Portal Release 1. Beginning with Oracle Portal 10*g* Release 1 (9.0.4), use OUTPUTPAGE.

Syntax OUTPUTFOLDER=*folder*

Values

folder Any valid folder name (*internal name*) used in Oracle WebDB Release 2.2 or Oracle9*i*AS 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 Oracle9*i*AS Portal Release 1.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.7.2 OUTPUTGRAPHFORMAT

Table A–73 indicates which components can use the OUTPUTGRAPHFORMAT keyword.

Table A–72 Components That Use OUTPUTGRAPHFORMAT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description Use OUTPUTGRAPHFORMAT to specify the GRAPH image format.

Syntax OUTPUTGRAPHFORMAT={SVG}

Values

- SVG (default)

  Supported Graph image format when DESFORMAT value is HTML or HTMLCSS.

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 outputgraphformat=SVG
```

```
rwrun report=images.rdf destype=file desformat=html desname=images.html
userid=user_id outputgraphformat=SVG
```

A.7.3 OUTPUTIMAGEFORMAT

Table A–73 indicates which components can use the OUTPUTIMAGEFORMAT keyword.

Table A–73 Components That Use OUTPUTIMAGEFORMAT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description Use OUTPUTIMAGEFORMAT to specify the format for images in report output.

Syntax OUTPUTIMAGEFORMAT={PNG|JPEG|JPG|GIF|BMP}

Values

- PNG,JPEG (default),JPG

  Supported image formats when DESFORMAT value is PDF, HTML, HTMLCSS, RTF, or SPREADSHEET.

- GIF

  Supported image format when DESFORMAT value is PDF, HTML, HTMLCSS, or SPREADSHEET.

- 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.
- A report containing images and run to DESFORMAT=SPREADSHEET using a secured Reports Server is not supported. This is due to Microsoft Excel's limitation on cookie support. Alternatively, you can write the Excel output from a secure Reports Server to a URL using WebDAV.

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.7.4 OUTPUTPAGE

Table A–74 indicates which components can use the OUTPUTPAGE keyword.

Table A–74 Components That Use OUTPUTPAGE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use OUTPUTPAGE to specify the name of the Oracle Portal page to which report output should be pushed. For backward compatibility with earlier versions (Oracle WebDB Release 2.2 and Oracle9*i*AS Portal Release 1), see OUTPUTFOLDER.

Syntax OUTPUTPAGE=*page*

Values

page Any valid page name (*internal name*) used in Oracle Portal.

Default Oracle_Reports_Output

Usage Notes

- Use of this keyword is *optional* to push Oracle Reports output to Oracle Portal:
  - If you do not specify an output page, Oracle 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 Oracle 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, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.7.5 OVERWRITE

Table A–75 indicates which components can use the OVERWRITE keyword.

Table A–75 Components That Use OVERWRITE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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.7.6 PARAMETER

Table A–75 indicates which components can use the *parameter* keyword.

Table A–76 Components That Use parameter

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	yes	yes	yes

Description Use *parameter* to specify a parameter that is part of the report's definition. The value you specify is used for that parameter's value.

Syntax *parameter=value*

Values

- *parameter* The name of the parameter as defined in the report.
- *value* Any valid value for that parameter.

Default Taken from the Initial Value property of the parameter.

Usage Notes

- You can override the default value on the Runtime Parameter Form.
- Values may be in single or double quotes. The effect of single or double quotes is operating-system specific.
- If you specify runfile(s) using the REPORT keyword, the parameter value(s) will be validated using the settings specified for each parameter (for example, Input Mask, Validation Trigger) in the report(s).
- If you do not specify runfile(s) using the REPORT keyword, the parameter value(s) are not parsed, or validated, because they may refer to parameters that have not yet been defined.

A.7.7 P_AVAILABILITY

Table A–77 indicates which components can use the P_AVAILABILITY keyword.

Table A–77 Components That Use P_AVAILABILITY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 Oracle Portal before running the SQL*PLUS script. If it does not, an invalid package may be created.

A.7.8 P_DESCRIPTION

Table A–78 indicates which components can use the P_DESCRIPTION keyword.

Table A–78 Components That Use P_DESCRIPTION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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.7.9 P_FORMATS

Table A–79 indicates which components can use the P_FORMATS keyword.

Table A–79 Components That Use P_FORMATS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 (for example, HTML) or a list of valid destination formats enclosed by parentheses with a comma separating the names (for example: HTMLCSS, PDF, RTF).

Usage Notes If the destination format for the report is DELIMITEDDATA, it may not be possible to batch register the report in Oracle Portal. As a workaround, you can define a different destination format, then batch register the report, and later manually edit the report to DESFORMAT=DELIMITEDDATA. For more information about batch registering reports, see Appendix C, "Batch Registering Reports in Oracle Portal"

Default None

A.7.10 P_JDBCPDS

Table A–80 indicates which components can use the P_JDBCPDS keyword.

Table A–80 Components That Use P_JDBCPDS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description Use P_JDBCPDS to specify the JDBC pluggable data source (PDS) connection string to connect to a database for running a report containing a JDBC query.

> **See Also:** Chapter 14, "Configuring and Using the Pluggable Data sources".

Syntax P_JDBCPDS=*userid*/*password*@*database*

Values A valid JDBC PDS connection string where:

- *userid* The user ID for connecting to the JDBC pluggable data source.
- *password* The password for the user ID.
- *database* The database connection information, specific to the particular data source, as detailed in Section 14.1.2.1, "Sample Connection Information".

Usage Notes P_JDBCPDS is the default sign-on name for connecting to a JDBC PDS. You can change this name in Reports Builder when defining the JDBC query in the Report Wizard. The new sign-on name can be used on the command line as the value for the P_JDBCPDS command line keyword. For more information on defining your JDBC query in Reports Builder, see Section 14.1.2, "Defining and Running a JDBC Query".

Example

To connect to a Sybase data source:

```
P_JDBCPDS=sybuser/sybpwd@server1.mydomain.com:1300
```

A.7.11 P_NAME

Table A–81 indicates which components can use the P_NAME keyword.

Table A–81 Components That Use P_NAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	no	no

Description Use P_NAME to specify the report name displayed in Oracle 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 Oracle Portal. This option is typically left blank.
- The report name cannot be prefaced with numeric characters (for example, 401K_report is an invalid file name and my_401K_report is valid).

A.7.12 P_OWNER

Table A–82 indicates which components can use the P_OWNER keyword.

Table A–82 Components That Use P_OWNER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	no	no

Description Use P_OWNER to specify the Oracle 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 Oracle Portal DB Provider name.

Default The name of the Oracle Portal DB Provider to which you are connected when you run the SQL*PLUS script file.

A.7.13 P_PFORMTEMPLATE

Table A–83 indicates the Components That Use the P_PFORMTEMPLATE keyword.

Table A–83 Components That Use P_PFORMTEMPLATE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	no	no

Description Use P_PFORMTEMPLATE to specify the name of the Oracle 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 Oracle Portal template name.

Default None

A.7.14 P_PRINTERS

Table A–84 indicates the Components That Use the P_PRINTERS keyword.

Table A–84 Components That Use P_PRINTERS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 (for example, PRT1), or a list of valid printers enclosed by parentheses with a comma separating the names (for example, (PRT1,PRT2,PRT3)).

Default None

Usage Note Access to the printer(s) should already exist in Oracle Portal before running the SQL*Plus script.

A.7.15 P_PRIVILEGE

Table A–85 indicates which components can use the P_PRIVILEGE keyword.

Table A–85 Components That Use P_PRIVILEGE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 Oracle Portal can recognize (for example, SCOTT), or a list of user names or roles enclosed by parentheses with a comma separating the names (for example, (SCOTT,JABERS,PMARTIN)).

Default None

A.7.16 P_SERVERS

Table A–86 indicates which components can use the P_SERVERS keyword.

Table A–86 Components That Use P_SERVERS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 (for example, repserver), or a list of valid Reports Server TNS names enclosed by parentheses with a comma separating the names (for example, (repserver, acct_server, sales_server)).

Default None

Usage Notes Access to the Reports Server(s) should already exist in Oracle Portal.

A.7.17 P_TRIGGER

Table A–87 indicates the Components That Use the P_TRIGGERS keyword.

Table A–87 Components That Use P_TRIGGER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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).

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

Example

```
P_TRIGGER="Is begin IF UPPER(DESTYPE) = ''PRINTER'' AND EMPNAME = ''SMITH'' THEN
```

```
RETURN(TRUE); ELSE RETURN(FALSE); END IF; end;"
```

A.7.18 P_TYPES

Table A–88 indicates which components can use the P_TYPES keyword.

Table A–88 Components That Use P_TYPES

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 (for example, CACHE to display in a browser), or a list of valid destination types enclosed by parentheses with a comma separating the names (for example, (CACHE,MAIL,PRINTER)).

Default None

A.7.19 PAGEGROUP

Table A–89 indicates which components can use the PAGEGROUP keyword.

Table A–89 Components That Use PAGEGROUP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use PAGEGROUP to specify the name of the Oracle Portal page group to which report output should be pushed. For backward compatibility with earlier releases, see SITENAME (for Oracle WebDB Release 2.2) and CONTENTAREA (for Oracle9*i*AS Portal Release 1). The page group must be created in Oracle Portal before you can use this keyword.

Syntax PAGEGROUP=*pagegroup*

Values

pagegroup Any valid page group name (*internal name*) used in Oracle Portal.

Default None

Usage Notes

- Use of this keyword is *required* to push Oracle Reports output to Oracle Portal.
- The page group name should be the internal name and *not* the display name. The internal name is used to uniquely identify the Oracle Portal page instance.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*, STATUSFOLDER*, STATUSPAGE.

* maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.7.20 PAGESIZE

Table A–90 indicates which components can use the PAGESIZE keyword.

Table A–90 Components That Use PAGESIZE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
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 Oracle 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 Oracle 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.7.21 PAGESTREAM

Table A–91 indicates which components can use the PAGESTREAM keyword.

Table A–91 Components That Use PAGESTREAM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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.7.22 PARAMFORM

Table A–92 indicates which components can use the PARAMFORM keyword.

Table A–92 Components That Use PARAMFORM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use PARAMFORM to specify whether to display the Runtime Parameter Form when you execute a report through Oracle Reports Servlet (rwservlet). PARAMFORM is used only to supply parameters to paper layout reports, not JSP-based Web 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 Oracle Portal environment. This is because Oracle Portal enables you to set up a Report Runtime Parameter Form, which would conflict with a Parameter Form you specify with the PARAMFORM keyword.

Example

```
http://myias.mycomp.com:7779/rwservlet?server=myrepserver+report=test.rdf
+userid=scott/tiger@mydb+destype=cache+desformat=htmlcss+paramform=html
```

A.7.23 PARSEQUERY

Table A–93 indicates which components can use the PARSEQUERY keyword.

Table A–93 Components That Use PARSEQUERY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use PARSEQUERY to parse an rwservlet query and display the constructed Reports Server command line.

Syntax http://*your_webserver*/reports/rwservlet/parsequery[?] [server=*server_name*][&authid=*username*/*password*]

Values See Syntax

Default None

Usage Notes

- This keyword is a command that does not require a value; that is, commands are entered by themselves without a corresponding value.
- Related keywords are SERVER and AUTHID.

A.7.24 PDFCOMP

Table A–94 indicates which components can use the PDFCOMP keyword.

Table A–94 Components That Use PDFCOMP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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.7.25 PDFEMBED

Table A–95 indicates which components can use the PDFEMBED keyword.

Table A–95 Components That Use PDFEMBED

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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.7.26 PDFOWNER

Table A–96 indicates which components can use the PDFOWNER keyword.

Table A–96 Components That Use PDFOWNER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Note: To generate PDF encrypted report output using Oracle Reports Builder (rwbuilder), you must pass at least one of the PDF encryption command line keywords (PDUSER, PDFOWNER, or PDFSECURITY) in the command to start rwbuilder.

In Oracle Reports Builder, select **Generate to File > PDF** to generate the PDF encrypted output.

Description Use PDFOWNER to specify the owner password for encrypted PDF report output. This password is used in combination with PDFUSER to specify authorization privileges for opening PDF report output (in Acrobat Reader 5.0 or later) and changing the passwords or access permissions of the PDF report output (in Acrobat Writer 6.0 or later). Refer to Table 11–4 for the effect of the possible combinations of the PDFOWNER and PDFUSER command line keywords.

See Section 11.1.4, "Encryption, Password Protection, and Permissions Security".

Syntax PDFOWNER=*owner_password*

Values

owner_password The owner password that must be entered by the end user when prompted.

Default None

A.7.27 PDFSECURITY

Table A–97 indicates which components can use the PDFSECURITY keyword.

Table A–97 Components That Use PDSECURITY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

> **Note:** To generate PDF encrypted report output using Oracle Reports Builder (rwbuilder), you must pass at least one of the PDF encryption command line keywords (PDUSER, PDFOWNER, or PDFSECURITY) in the command to start rwbuilder.
>
> In Oracle Reports Builder, select **Generate to File > PDF** to generate the PDF encrypted output.

Description Use PDFSECURITY to suppress up to 8 permissions for encrypted PDF report output. PDFSECURITY may be specified with or without PDFUSER and PDFOWNER.

See Section 11.1.4, "Encryption, Password Protection, and Permissions Security".

Syntax PDFSECURITY=*suppress_permission*[,*suppress_permission*...]

Values

suppress_permission Any of the values listed in Table A–98, which specifies the associated effect when the value is specified:

Table A–98 PDFSECURITY Values

Value	Effect
NOHIRESPRINTING	Disallows high resolution printing; instead, the document prints at a low resolution.
NOMODIFYCONTENTS	Disallows modifying the contents of the document by operations other than those controlled by NOMODIFYANNOTATIONS, NOFILLIN, and NOASSEMBLY.
NOCOPY	Disallows copying or otherwise extracting text and graphics from the document by operations other than that controlled by NOSCREENREADERS.
NOMODIFYANNOTATIONS	Disallows adding or modifying text annotations, fill in interactive form fields, and, if NOMODIFYCONTENTS is also specified, disallow creating or modifying interactive form fields (including signature fields).
NOFILLIN	Disallows filling in existing interactive form fields (including signature fields), even if NOMODIFYANNOTATIONS is not specified.
NOSCREENREADERS	Disallows extracting text and graphics (for accessibility support or for other purposes).
NOASSEMBLY	Disallows assembling the document (insert, rotate, or delete pages and create bookmarks or thumbnail images), even if NOMODIFYCONTENTS is not specified.
NOPRINTING	Disallows printing the PDF document.

Default None (all permissions are granted).

Example

```
PDFSECURITY=NOHIRESPRINTING,NOMODIFYCONTENTS,NOCOPY
```

A.7.28 PDFUSER

Table A–99 indicates which components can use the PDFUSER keyword.

Table A–99 Components That Use PDFUSER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Note: To generate PDF encrypted report output using Oracle Reports Builder (rwbuilder), you must pass at least one of the PDF encryption command line keywords (PDUSER, PDFOWNER, or PDFSECURITY) in the command to start rwbuilder.

In Oracle Reports Builder, select **Generate to File > PDF** to generate the PDF encrypted output.

Description Use PDFUSER to specify the user password for encrypted PDF report output. This password is used in combination with PDFOWNER to specify authorization privileges for opening PDF report output (in Acrobat Reader 5.0 or later) and changing the passwords or access permissions of the PDF report output (in Acrobat Writer 6.0 or later). Refer to Table 11–4 for the effect of the possible combinations of the PDFOWNER and PDFUSER command line keywords.

See Section 11.1.4, "Encryption, Password Protection, and Permissions Security".

Syntax PDFUSER=*user_password*

Values

user_password The user password that must be entered by the end user when prompted.

Default None

A.7.29 PFACTION

Table A–100 indicates which components can use the PFACTION keyword.

Table A–100 Components That Use PFACTION

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	no	no

Note: PFACTION is also used with the Oracle Forms Services keyword RUN_REPORT_OBJECT, which is the most secure approach for calling Oracle Reports from Oracle Forms on the Web. For detailed information about using RUN_REPORT_OBJECT to call Oracle Reports from Oracle Forms, refer to the *Oracle Application Server 10g Integrating Oracle Reports in Oracle Forms Services Applications* white paper on OTN (http://otn.oracle.com/products/forms/pdf/10g/frm10gsrw10g.pdf). Also refer to the *Forms Services Deployment Guide*.

Description Use PFACTION to specify the action string for a Parameter Form. When a request is submitted through rwclient or the Oracle Forms Services RUN_REPORT_OBJECT keyword to run a report that has a Parameter Form, the Parameter Form generated cannot be used unless PFACTION is included in the command line. The unusable Parameter Form is caused by an empty action attribute (because rwclient and RUN_REPORT_OBJECT call Oracle Reports directly on the server, Oracle Reports cannot access the Web environment to obtain the information required to populate the action attribute when generating the HTML Parameter Form).

The action attribute is part of the standard HTML form tag that defines the action that is performed when the end user clicks **Submit**. The action attribute in the Oracle Reports Parameter Form should contain hidden runtime parameters that are required to process the Oracle Reports request after the end user clicks **Submit**.

Syntax PFACTION=*request_URL_to_rwservlet*?_hidden_*encoded_original_url_string*

Values

- *request_URL_to_rwservlet* The http://*host*:*port*/reports/rwservlet URL.
- *encoded_original_url_string* The query string to run the report.

Example

```
rwclient.exe report=pform.rdf destype=cache desformat=pdf paramform=yes
PFACTION="http://mymachine.mycompany.com:7777/reports/rwservlet?_hidden_
report=pform.rdf%20destype=cache%20desformat=pdf"
```

A.7.30 PRINTJOB

Table A–101 indicates which components can use the PRINTJOB keyword.

Table A–101 Components That Use PRINTJOB

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	yes	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 YES

Usage Notes

- When a report is run as a spawned process (that is, one component, such as rwrun, is called from within another component, 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.7.31 READONLY

Table A–102 indicates which components can use the READONLY keyword.

Table A–102 Components That Use READONLY

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description Use READONLY to request read consistency across multiple queries in a report. When accessing data from an Oracle database, read consistency is accomplished by a SET TRANSACTION READ ONLY statement (refer to your Oracle database documentation for more information on SET TRANSACTION READ ONLY).

> **Note:** For more information on SET TRANSACTION READ ONLY, refer to the Oracle SQL documentation (available on the Oracle Technology Network, (http://www.oracle.com/technetwork/developer-tools/sql-developer/documentation/index.html).

Syntax READONLY={YES|NO}

Values

- YES Requests read consistency.
- NO Do not provide read consistency.

Default NO

Usage Notes

- READONLY is only useful for reports using multiple queries. An Oracle database 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.7.32 RECURSIVE_LOAD

Table A–103 indicates which components can use the RECURSIVE_LOAD keyword.

Table A–103 Components That Use RECURSIVE_LOAD

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	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.7.33 REPLYTO

Table A–104 indicates which components can use the REPLYTO keyword.

Table A–104 Components That Use REPLYTO

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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, and SUBJECT. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.7.34 REPORT | MODULE

See MODULE | REPORT.

A.7.35 ROLE

Table A–105 indicates which components can use the ROLE keyword.

Table A–105 Components That Use ROLE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	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.8 Command Line Keywords (RUNDEBUG to WEBSERVER_PORT)

This section provides a brief description of the Oracle Reports components and the keywords that each component can use.

A.8.1 RUNDEBUG

Table A–106 indicates which components can use the RUNDEBUG keyword.

Table A–106 Components That Use RUNDEBUG

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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.8.2 SAVE_RDF

Table A–107 indicates which components can use the SAVE_RDF keyword.

Table A–107 Components That Use SAVE_RDF

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	yes	yes	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 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.8.3 SCHEDULE

Table A–108 indicates which components can use the SCHEDULE keyword.

Table A–108 Components That Use SCHEDULE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	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 (that is, 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.

Syntax SCHEDULE=*string*

where *string* is:

[*FREQ* from] *TIME* [retry {*n*} after *LEN* expires {on|after} time|*n*]

Table A–109 lists and explains the values used in this string.

Table A–109 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

Examples

```
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"
```

A.8.4 SERVER

Table A–110 indicates which components can use the `SERVER` keyword.

Table A–110 Components That Use SERVER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	yes	yes

Description Use `SERVER` to specify the name of the Reports Server you want to use to run this report.

Syntax `SERVER=`*`server_name`*

Values See Syntax

Usage Notes

- For jobs run with `rwservlet` or as a JSP, you can omit the `SERVER` keyword if you have specified a default server in the Oracle Reports Servlet (`rwservlet`) 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.8.5 SHOWAUTH

Table A–111 indicates which components can use the SHOWAUTH keyword.

Table A–111 Components That Use SHOWAUTH

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use SHOWAUTH to display the Reports Server logon page and run the report.

Syntax `http://your_webserver/reports/rwservlet/showauth[?]`
`[server=server_name][&authid=username/password]`
`[&nextpage=key_in_cgicmd.dat]&authtype={s|d}`

Values See Syntax

Default None

Usage Notes

- This keyword is a command that does not require a value; that is, commands are entered by themselves without a corresponding value.
- After authentication, the URL specified by the cgicmd.dat key in the nextpage parameter will be run by the Reports servlet.
- The parameter authtype is mandatory.

  When authtype=s the Reports System User Authentication dialog box is displayed. When authtype=d, the Reports Database User Authentication dialog box is displayed.
- Related keywords are SERVER and AUTHID.

A.8.6 SHOWENV

Table A–112 indicates which components can use the SHOWENV keyword.

Table A–112 Components That Use SHOWENV

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use SHOWENV to display the rwservlet configuration file (rwservlet.properties).

Syntax `http://your_webserver/reports/rwservlet/showenv[?]`
`[server=server_name][&authid=username/password]`

Values See Syntax

Default None

Usage Notes

- This keyword is a command that does not require a value; that is, commands are entered by themselves without a corresponding value.

- Related keywords are SERVER and AUTHID.

A.8.7 SHOWJOBID

Table A–113 indicates which components can use the SHOWJOBID keyword.

Table A–113 Components That Use SHOWJOBID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description SHOWJOBID shows the status of the Reports Server job with job ID *n*.

Syntax http://*your_webserver*/reports/rwservlet/showjobidn[?]
[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; that is, 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.

A.8.8 SHOWJOBS

Table A–114 indicates which components can use the SHOWJOBS keyword.

Table A–114 Components That Use SHOWJOBS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use SHOWJOBS to display a Web view of Reports Server queue status. In a cluster environment, SHOWJOBS displays all the in-process servers that are part of the cluster. You can view the jobs of both cluster and individual servers. Even if the server is down, you can retrieve information about past jobs for that server.

Syntax http://*your_webserver*/*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 does not require a value; that is, keywords are entered by themselves without a corresponding value.
- The name of the Reports Server must be specified implicitly by environment variable or Oracle Reports Servlet (`rwservlet`) 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).
- Related keywords are SERVER, AUTHID, and STATUSFORMAT.

Example

Use `SHOWJOBS` to display a consolidated job queue in a high availability environment, as follows:

`http://`*`host`*`:`*`port`*`/reports/rwservlet/showjobs`

where *host* and *port* is the load balancer host name and port number.

A.8.9 SHOWMAP

Table A–115 indicates which components can use the `SHOWMAP` keyword.

Table A–115 Components That Use SHOWMAP

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use `SHOWMAP` to display `rwservlet` key mappings.

Syntax `http://`*`your_webserver`*`/reports/rwservlet/showmap[?]`
`[server=`*`server_name`*`][&authid=`*`username/password`*`]`

Values See Syntax

Default None

Usage Notes:

- This keyword does not require a value; that is, commands are entered by themselves without a corresponding value.
- Related keywords are SERVER and AUTHID.

A.8.10 SHOWMYJOBS

Table A–116 indicates which components can use the `SHOWMYJOBS` keyword.

Table A–116 Components That Use SHOWMYJOBS

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use SHOWMYJOBS to display the Reports Server queue status for a particular user.

Syntax http://*your_webserver*/reports/rwservlet/showmyjobs[?]
[server=*server_name*][&authid=*username*/*password*]
[&statusformat={html|xml|xmldtd}]

Values See Syntax

Default None

Usage Notes

- This keyword does not require a value; that is, 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.

A.8.11 SHUTDOWN

Table A–117 indicates which components can use the SHUTDOWN keyword.

Table A–117 Components That Use SHUTDOWN

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver	rwbridge
no	no	no	no	no	yes	yes

Description Use SHUTDOWN to shut down a previously running server, or to shut down the Oracle Reports Bridge.

When used with rwserver, you must also use AUTHID to supply a user name and password.

When used with rwbridge, you must also use AUTHID to supply a user name and password if the Oracle Reports Bridge is secured (the identifier element is set in the Oracle Reports Bridge configuration file).

Syntax SHUTDOWN={NORMAL|IMMEDIATE}

Values

- NORMAL Shuts the server or Oracle Reports Bridge down gracefully, using normal shutdown procedures.
- IMMEDIATE Shuts the server or Oracle Reports Bridge down immediately, without waiting for other processes to complete running.

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.8.12 SITENAME

Table A–118 indicates which components can use the SITENAME keyword.

Table A–118 Components That Use SITENAME

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use SITENAME to specify the name of the Oracle WebDB Release 2.2 site to which report output should be pushed. This keyword is maintained for backward compatibility with Oracle WebDB Release 2.2; for backward compatibility with Oracle9*i*AS Portal Release 1, see CONTENTAREA. Beginning with Oracle Portal 10*g* Release 1 (9.0.4), 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 Oracle WebDB Release 2.2.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.8.13 SOURCE

Table A–119 indicates which components can use the SOURCE keyword.

Table A–119 Components That Use SOURCE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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 (for example, (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 (that is, 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 Oracle 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 string is prefixed to the report/library name, the USERID string is assumed to be that 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.8.14 SQLTRACE

Table A–120 indicates which components can use the SQLTRACE keyword.

Table A–120 Components That Use SQLTRACE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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.8.15 SSOCONN

Table A–121 indicates which components can use the SSOCONN keyword.

Table A–121 Components That Use SSOCONN

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	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_string_parameter*]][,*key*[/*type*[/*conn_string_parameter*]]]

Values The following information describes the variable values expressed in the SSOCONN syntax:

- *key* refers to a connection string value stored in Oracle Internet Directory.
- *type* is the kind of data source to which you are connecting, to identify the format in the string associated with *key*. The *type* value 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)
- *conn_string_parameter* is the name of the Oracle Reports system or user parameter to be used to pass the connection string to rwservlet to run the report. For example, in the case of the OracleDB data source, Oracle Reports receives the connection string through the USERID parameter and uses it to connect to the specified Oracle database. Similarly, for JDBCPDS, P_JDBCPDS is used. If you have your own custom pluggable data sources, you must define your own user parameter for passing the connection string to Oracle Reports and specify it as *conn_string_parameter* for SSOCONN.

For example:

```
SSOCONN=mykey/OracleDB/USERID
```

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
  ```

- 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=NO in rwservlet.properties.
- Simplified versions of SSOCONN are also available, as shown in Table A–122.

Table A–122 Simplified Versions of the SSOCONN Option

Option	Description
ssoconn=mkey	When only the key name is specified, the default values for *type* and *conn_string_parameter* are: OracleDB and USERID.
ssoconn=mkey/PDSApp	When both key name and data source type are specified, the default value for *conn_string_parameter* is USERID.

- 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 on Oracle Reports and Single Sign-On (SSO), see Chapter 17, "Configuring and Administering Oracle Single Sign-On".

A.8.16 STATUSFOLDER

Table A–123 indicates which components can use the STATUSFOLDER keyword.

Table A–123 Components That Use STATUSFOLDER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use STATUSFOLDER to specify the name of the Oracle WebDB Release 2.2 or Oracle9*i*AS Portal Release 1 folder to which job status information should be pushed. 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 Oracle9*i*AS Portal Release 1. Beginning with Oracle Portal 10*g* Release 1 (9.0.4), use STATUSPAGE.

Syntax STATUSFOLDER=*folder*

Values

folder Any valid folder name (*internal name*) used in Oracle WebDB Release 2.2 or Oracle9*i*AS 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 Oracle9*i*AS Portal Release 1.
- The value for this keyword is case sensitive.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.8.17 STATUSFORMAT

Table A–124 indicates which components can use the STATUSFORMAT keyword.

Table A–124 Components That Use STATUSFORMAT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	no	yes	no

Description Use STATUSFORMAT to specify the format for the Web view of the Reports Server queue status.

Syntax http://*yourwebserver*/rwservlet/showjobs?server=*server_name*&statusformat={html|xml|xmldtd}

Values

- html Outputs the Reports Server queue status in HTML format.
- xml Outputs the Reports Server queue status in XML format.

> **Note:** Oracle Reports includes a param named ERRORSTATUSXML that can be used to view the error status in the XML format even if Reports job fails. To view the ERRORSTATUSXML in XML, set ERRORSTATUSXML=Yes on the rwservlet command line.
>
> In case of event-based reporting, before calling the srw.run_reports you must complete the following steps to enable ERRORSTATUSXML:
>
> - Call srw.ignore_server_error, and
> - Call srw.add_parameter to add errorstatusxml=yes to the request.

- xmldtd Outputs the Reports Server queue status 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.8.18 STATUSPAGE

Table A–125 indicates which components can use the STATUSPAGE keyword.

Table A–125 Components That Use STATUSPAGE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use STATUSPAGE to specify the name of the Oracle Portal page to which job status information should be pushed. If this is omitted, a new page is created called Oracle_Reports_Status. For backward compatibility with earlier releases (Oracle WebDB Release 2.2 and Oracle9*i*AS Portal Release 1), see STATUSFOLDER.

Syntax STATUSPAGE=*page*

Values

page Any valid page name (*internal name*) used in Oracle Portal.

Default Oracle_Reports_Status

Usage Notes

- Use of this keyword is *optional* to push output to Oracle Portal.

- The value for this keyword is case sensitive.
- Relevant keywords include CONTENTAREA*, EXPIREDAYS, ITEMTITLE, OUTPUTFOLDER*, OUTPUTPAGE, PAGEGROUP, SITENAME*, STATUSFOLDER*, STATUSPAGE.

  * maintained for backward compatibility with Oracle9*i*AS Portal Release 1 and Oracle WebDB Release 2.2.

A.8.19 STYPE

Table A–126 indicates which components can use the STYPE keyword.

Table A–126 Components That Use STYPE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	no	yes	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.8.20 SUBJECT

Table A–127 indicates which components can use the SUBJECT keyword.

Table A–127 Components That Use SUBJECT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	no	no	yes	no

Description Use SUBJECT to specify the subject line of an e-mail.

Syntax SUBJECT="*string*"

Values Any text string.

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, and REPLYTO. Note that DESNAME is used to specify the main recipient(s) of the e-mail.

A.8.21 SUPPRESSLAYOUT

Table A–128 indicates which components can use the SUPPRESSLAYOUT keyword.

Table A–128 Components That Use SUPPRESSLAYOUT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	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 through 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.8.22 TOLERANCE

Table A–129 indicates which components can be used with the TOLERANCE keyword.

Table A–129 Components That Use TOLERANCE

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	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 time zone 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 Oracle Reports Services reruns the report even if a duplicate report is found in the cache.
- If a report is being processed (that is, in the current job queue) when an identical job is submitted, then Oracle Reports Services reuses the output of the currently running job even if TOLERANCE is not specified or is set to zero.

A.8.23 URLPARAMETER

Table A–130 indicates which components can use the URLPARAMETER keyword.

Table A–130 Components That Use URLPARAMETER

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	yes	no

Description Use URLPARAMETER to specify the URL that is to be fetched with the URL engine.

Syntax URLPARAMETER=http://*your_webserver*/*page_name*.html

Values Any valid URL.

Default None

Usage Notes This keyword is relevant when jobType=rwurl in the job element in the Reports Server configuration file, and a URL engine is in place.

A.8.24 USEJVM

Table A–131 indicates which components can use the USEJVM keyword.

Table A–131 Components That Use USEJVM

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	no	no	no	no	no

Description Use USEJVM to specify whether or not rwclient should use Java Virtual Machine (JVM) to communicate with Reports Server.

Syntax USEJVM=YES|NO

Values

- YES rwclient starts JVM and tries to connect to Reports Server using CORBA; if this fails, then it will attempt to connect using SQLNet, which is available for backward compatibility.
- NO rwclient does not start JVM; instead, it uses SQLNet to communicate with Reports Server (Oracle Reports 6*i* Server).

Default YES

A.8.25 USERID

Table A–132 indicates which components can use the USERID keyword.

Table A–132 Components That Use USERID

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	yes	yes	no

Description Use USERID only if you are *not* using Single Sign-On 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. rwclient does not do round trips to the reports server twice when password is not entered in USERID= parameter, so enter password prompt would not be displayed.It is TRUE for both rwrun and rwservlet.

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 cgicmd.dat key map file (see Section 18.14, "Using a Key Map File") to specify the runtime command, and include the %D option in the relevant key mapping entry.

Syntax userid=*username*[/*password*][@*database*]

Values

- *username* Username assigned by the database administrator.
- *password* Password for the username. See "Usage Notes", below.
- *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.
- It is strongly recommended that you do not include the password when using USERID with rwbuilder, rwrun, rwclient, or rwconverter. On many operating systems, this information can become available to any user (for example, with the ps command on UNIX). Instead, use the SSOCONN keyword.

A.8.26 USERSTYLES

Table A–133 indicates which components can use the USERSTYLES keyword.

Table A–133 Components That Use USERSTYLES

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
yes	yes	yes	no	yes	no

Description Use USERSTYLES to specify whether an external style sheet file (.css) is associated with a report when generating HTMLCSS output. The style sheets to be applied to the report are specified by the report's Style Sheets property. The value is set to YES by default, and will override any design-time styles included in the Paper Design layout.

> **See Also:** *Oracle Reports User's Guide to Building Reports* for more information on online HTML formatting.

Syntax USERSTYLES=YES|NO

Values

- YES Associates your report with one or more external style sheets, as specified by the report's Style Sheets property, when generating HTMLCSS output.
- NO Associates your report with the formatting applied during the design of the report. External style sheets will be ignored.

Default YES

Usage Notes

- If you specify a value other than YES or NO, Oracle Reports defaults to YES.
- If you find that your report is not associated with an external style sheet, ensure the following:
  - You have specified the correct path to the style sheet in the Style Sheets property.
  - The styles specified by the CSS Class Name and the CSS ID properties are defined in the specified style sheets.

    > **See Also:** *Oracle Reports online Help* for more information on the Style Sheets, CSS Class Name, and CSS ID properties.

  - USERSTYLES is set to YES.

Example

```
http://myias.mycomp.com:7779/reports/rwservlet?server=myrepserv+report=test.jsp+us
erid=scott/tiger@mydb+desformat=HTMLCSS+DESTYPE=cache+userstyles=yes
```

A.8.27 VALIDATETAG

Table A–133 indicates which components can use the VALIDATETAG keyword.

Table A–134 Components That Use VALIDATETAG

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	yes	no	no	no

Description VALIDATETAG specifies whether to enforce JSP tag validation and check for items such as duplicate field identification or malformed attributes when designing or deploying a JSP-based Web report.

> **See Also:** Section 24.7, "Running the Report" for more information about using VALIDATETAG to tune the performance of your report.

Syntax VALIDATETAG=YES|NO

Values

- YES Enforces tag validation and checks for items such as duplicate field identification or malformed attributes.
- NO Turns tag validation off.

Default

- YES At design time, when running a JSP-based Web report from Oracle Reports Builder.
- NO At run time, when deploying a JSP-based Web report.

Usage notes

- This feature is useful only during the design phase, but not in the production environment. By default, VALIDATETAG=YES in Oracle Reports Builder during report design, and VALIDATETAG=NO in Oracle Reports Services for report deployment. To turn this option on when deploying a report, specify VALIDATETAG=YES in your http request (for example, http://my.server.com/myreport.jsp?VALIDATETAG=YES).
- Using VALIDATETAG=YES when deploying a report slows performance.
- If you start Reports Builder from the command line with rwbuilder VALIDATETAG=NO, you run the risk of designing a report with invalid JSP tag structure.

A.8.28 WEBSERVER_DEBUG

Table A–135 indicates which components can use the WEBSERVER_DEBUG keyword.

Table A–135 Components That Use WEBSERVER_DEBUG

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	yes	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 Creates debugging files.
- NO Does 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.8.29 WEBSERVER_DOCROOT

Table A–136 indicates which components can use the WEBSERVER_DOCROOT keyword.

Table A–136 Components That Use WEBSERVER_DOCROOT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	yes	no	no	no

Description Use WEBSERVER_DOCROOT to set the Oracle Reports 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*

Values The root folder directory for your report files.

Default $DOMAIN_HOME/servers/WLS_REPORTS/tmp/_WL_user/reports_<version>/<random_string>/war

Usage Notes

- Use this keyword only when you're running a job as a JSP.
- Relevant keywords include WEBSERVER_DEBUG, WEBSERVER_DOCROOT, WEBSERVER_PORT.

Example

```
WEBSERVER_DOCROOT=c:/temp/docroot
```

A.8.30 WEBSERVER_PORT

Table A–137 indicates which components can use the WEBSERVER_PORT keyword.

Table A–137 Components That Use WEBSERVER_PORT

rwclient	rwrun	rwbuilder	rwconverter	rwservlet	rwserver
no	no	yes	no	no	no

Description Use WEBSERVER_PORT to specify the port number an internal Web server listens to. Oracle Reports Builder runs the JSP reports at the port number specified by WEBSERVER_PORT.

Syntax `WEBSERVER_PORT=port_num`

Values

port_num Any valid port number

Default

Port number of the external web server such as Oracle HTTP Server

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

Environment Variables

Environment variables are parameters that configure the Oracle Reports environment. The Oracle Fusion Middleware installer automatically defines default values for relevant environment variables. Edit the environment variable settings to change the default behavior:

- On Windows, edit the environment variables through the Registry Editor (**Start** > **Run** > **Regedit**).
- On UNIX, edit the environment variables by revising and running the shell script that defines the initial default values (`reports.sh`). If you do this, keep a backup of the original, unaltered `reports.sh` file.

> **Note:** The `reports.sh` file in Oracle Reports 12*c* Release (12.2.1.3) may contain some required changes for the release. Thus, if you have made any changes to the `reports.sh` file in prior releases, save a backup before you perform your upgrade. Post-upgrade, merge your modifications with the `reports.sh` file installed with Oracle Reports12*c* Release (12.2.1.3).

If you set any environment variable in the `$DOMAIN_HOME/reports/bin/reports.sh` file (for UNIX) or in the registry file (for Windows), all the Reports Servers present in that Instance Home are affected.

If you want to set the environment variable only for a particular server, you must set the environment variable in the `<environment>` element in the `reports_process.xml` file of that particular server.

To set the environment variable for an In-process server, you must modify the settings in the `setStartupEnv.sh` file located at `$DOMAIN_HOME/bin`.

B.1 Environment Variables

Table B–1 provides an alphabetical summary list of all the Oracle Reports environment variables alphabetically, showing valid and default value.

- For more information on the globalization support environment variables, refer to Chapter 23, "Implementing Globalization and Bidirectional Support".
- The environment variables shown in italics are supported 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.

Table B–1 provides an alphabetical summary list of all the Oracle Reports environment variables alphabetically, showing valid and default value.

Table B–1 Oracle Reports Environment Variables

Keywords	Valid Values	Default
CA_GPREFS	Any directory on any drive	*ORACLE_HOME*
CA_UPREFS	Any directory on any drive	$DOMAIN_HOME\config\fmwconfig\components\ReportsToolsComponent\<reports_tools_name>\tools
DELIMITED_LINE_END	YES|NO	YES
DOC	Any directory on any drive	*ORACLE_HOME*\tools\doc
DEVELOPER_NLS_LANG	See Chapter 23, "Implementing Globalization and Bidirectional Support"	See Chapter 23, "Implementing Globalization and Bidirectional Support"
NLS_CALENDAR		
NLS_CREDIT		
NLS_CURRENCY		
NLS_DATE_FORMAT		
NLS_DATE_LANGUAGE		
NLS_DEBIT		
NLS_ISO_CURRENCY		
NLS_LANG		AMERICAN_AMERICA.WE8ISO8859P1
NLS_LIST_SEPARATOR		
NLS_MONETARY_CHARACTERS		
NLS_NUMERIC_CHARACTERS		
NLS_SORT		
ORACLE_AFM	Any directory on any drive	Not defined
ORACLE_HOME	Any directory on any drive	C:\orawin
ORACLE_HPD	Any directory on any drive	Not defined
ORACLE_PATH	Any directory on any drive	Not defined
ORACLE_PPD	Any directory on any drive	Not defined
ORACLE_TFM	Any directory on any drive	Not defined
ORAINFONAV_DOCPATH	Any directory on any drive	Not defined
PRINTER	Name of default printer	Not defined
REMOTE	Any valid SQL* Net driver prefix and parameters	Not defined
REPORTS_ADD_HWMARGIN	YES|NO	NO
REPORTS_ALLOW_DB_CONNECT_STRING	YES|NO	NO
REPORTS_ARABIC_NUMERAL	ARABIC|HINDI|CONTEXT	ARABIC (Indo-Arabic)
REPORTS_BIDI_ALGORITHM	ORACLE|UNICODE|ENHANCED	ORACLE
REPORTS_CGIDIAGBODYTAGS	Any valid HTML attributes for the <BODY> tag.	Not defined

Table B–1 (Cont.) Oracle Reports Environment Variables

Keywords	Valid Values	Default
REPORTS_CGIDIAGHEADTAGS	Any HTML tags that are valid between <HEAD> and </HEAD>.	Not defined
REPORTS_CGIHELP	Any valid URL to a Web page or a HTML file.	A default help screen is displayed in the browser.
REPORTS_CGIMAP	A valid path to the key map file (see Section 18.14, "Using a Key Map File").	$DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/cgicmdd.dat
REPORTS_CGINODIAG	YES|NO	NO
REPORTS_CLASSPATH	The default values are mandatory. If any of the entries are removed, the Oracle Reports components may not behave correctly. Any additional user-defined directory or JAR file that contains Java Classes may be appended to the path.	%ORACLE_HOME%\reports\jlib\rwbuilder.jar; %ORACLE_HOME%\reports\jlib\rwrun.jar; %ORACLE_HOME%\jlib\zrclient.jar;
REPORTS_CONTAINSHTMLTAGS	YES|NO	YES
REPORTS_COOKIE_EXPIRE	Any number of minutes	30
REPORTS_CUPS_PRINTING	YES|NO	NO
REPORTS_DB_AUTH	Any HTML file that contains special authentication actions. It is recommended that you keep the default.	dbauth.htm
REPORTS_DEFAULT_DISPLAY	YES|NO	YES
REPORTS_DEFAULT_PIXEL_SIZE	Any value ranging from 72 through 200.	Surface resolution determined by Oracle Reports.
REPORTS_ENABLE_RTF_SPACING	YES|NO	NO
REPORTS_ENCRYPTION_KEY	Any encryption key.	reports9.0
REPORTS_ENHANCED_BIDIHANDLING	YES|NO	NO
REPORTS_ENHANCED_FONTHANDLING	YES|NO	YES
REPORTS_ENHANCED_SUBSET	YES|NO	YES
REPORTS_FONT_DIRECTORY	Any directory on any drive.	$DOMAIN_HOME/reports/fonts
REPORTS_GRAPH_IMAGE_DPI	72 through 300	72
REPORTS_IGNORE_IMAGE_TAG_RES	YES|NO	NO
REPORTS_IGNORE_SET_ROLE_ERROR	YES|NO	NO
REPORTS_JPEG_QUALITY_FACTOR	1 through 100	100
REPORTS_JVM_OPTIONS	List of JVM options in the JVM command line syntax.	-Xmx256M
REPORTS_NETWORK_CONFIG	Valid custom network configuration file name in ${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/	rwnetwork.conf
REPORTS_NLS_XML_CHARSETS	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*]...	Not defined.

Table B–1 (Cont.) Oracle Reports Environment Variables

Keywords	Valid Values	Default
REPORTS_NO_DUMMY_PRINTER	TRUE | not set	TRUE
REPORTS_NO_HTML_SPACE_REPLACE	YES | not set	not set
REPORTS_OUTPUTIMAGEFORMAT	GIF|JPEG|JPG|PNG|BMP	JPEG
REPORTS_PATH	Any directory on any drive.	%ORACLE_HOME%\REPORT\DEMO; %ORACLE_HOME%\REPORT\DEMO\BITMAP %ORACLE_HOME%\REPORT\DEMO\REQFILES
REPORTS_RESOURCE	Any directory on any drive.	%ORACLE_HOME%\reports\res\US
REPORTS_SERVER	Any Reports Server service entry name.	
REPORTS_SOLARIS_9	YES|NO	YES on Solaris 2.9; NO on other platforms
REPORTS_SPACE_BREAK	YES|NO	YES
REPORTS_SRWRUN_TO_SERVER	YES | not set	not set
REPORTS_SSLPORT	Any valid port number.	not set
REPORTS_SYS_AUTH	Any HTML file that contains special authentication actions. It is recommended that you keep the default.	sysauth.htm
REPORTS_TAGLIB_URI	Any "uri" that references the Oracle Reports tag library.	/WEB-INF/lib/reports_tld.jar
REPORTS_TMP	Any directory on any drive.	Windows: C:\Documents and Settings\<username>\Local Settings\Temp UNIX: /tmp
REPORTS_UTF8_XMLOUTPUT	YES|NO	YES
REPORTS_USEREXITS	Any user exit dynamic link library (along with its absolute path).	Not defined.
RW	A valid directory name.	%ORACLE_HOME\reports (Windows) $ORACLE_HOME/reports (UNIX)
TK_PRINT	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 **Solaris:** lpr -P'%n' -#%c -s	Not defined.

Table B–1 (Cont.) Oracle Reports Environment Variables

Keywords	Valid Values	Default
TK_PRINT_STATUS	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	Not defined.
TK_PRINTER	Name of default printer.	Not defined.
TK_AFM	Any directory on any drive.	Not defined.
TK_HPD	Any directory on any drive.	Not defined.
TK_PPD	Any directory on any drive.	Not defined.
TK_TFM	Any directory on any drive.	Not defined.
TNS_ADMIN	Any directory of any drive that has tnsnames.ora file in it.	DOMAIN_HOME/config/fmwconfig/
USERNAME	Any valid Oracle username (without the OPS$ prefix).	Not defined.
USER_NLS_LANG		

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:\orawin

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 through **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:\orawin`

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:\myreports_1012\tools\doc`

B.1.5 DEVELOPER_NLS_LANG

Description This environment variable specifies the language for the report. Chapter 23, "Implementing Globalization and Bidirectional Support" contains additional detailed information about this environment variable, including a table of valid values.

B.1.6 NLS_CALENDAR

Description This environment variable specifies the calendar system used.

B.1.7 NLS_CREDIT

Description This environment variable specifies the string used to indicate a positive monetary value.

B.1.8 NLS_CURRENCY

Description This environment variable specifies the local currency symbol.

B.1.9 NLS_DATE_FORMAT

Description This environment variable specifies the default format used for dates.

B.1.10 NLS_DATE_LANGUAGE

Description This environment variable specifies the default language used for dates.

B.1.11 NLS_DEBIT

Description This environment variable specifies the string used to indicate a negative monetary value.

B.1.12 NLS_ISO_CURRENCY

Description This environment variable specifies the ISO currency symbol.

B.1.13 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 23, "Implementing Globalization and Bidirectional Support" contains additional detailed information about this environment variable, including a table of valid values.

Syntax `NLS_LANG=` *`language_territory.charset`*

Valid Values

- *`language`* Specifies the language and its conventions for displaying messages and day and month names.
- *`territory`* Specifies the territory and its conventions for calculating week and day numbers.
- *`charset`* Specifies the character set used for the `UPPER`, `LOWER`, and `INITCAP` functions, and the type of sort used by an `ORDER BY` query. This argument also controls the character set used for displaying messages.

Default `AMERICAN_AMERICA.WE8ISO8859P1`

Usage Notes

- To change locales, you must modify this environment variable, in addition to the REPORTS_RESOURCE environment variable.
- On UNIX, you must set this variable in `$DOMAIN_HOME/reports/bin/reports.sh`. If this variable is already set in an environment where Oracle Reports is running,

then the value set in this environment is used instead of the value set in reports.sh. To use the value that is set in reports.sh, you must reset the value of the variable in the environment.

Examples 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.WE8ISO8859P1
```

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.WE8ISO8859P1
```

More examples:

```
NLS_LANG=Norwegian_Norway.NDK7DEC

NLS_LANG=Norwegian_Norway.WE8ISO8859P1

NLS_LANG=Japanese_Japan.JA16SJIS

NLS_LANG=Arabic_Egypt.AR8MSWIN1256

NLS_LANG=American_America.AR8MSWIN1256

NLS_LANG=American_America.WE8ISO8859P1
```

B.1.14 NLS_LIST_SEPARATOR

Description This environment variable specifies the character used to separate items in a list.

B.1.15 NLS_MONETARY_CHARACTERS

Description This environment variable specifies the decimal character and thousands separator for monetary values.

B.1.16 NLS_NUMERIC_CHARACTERS

Description This environment variable specifies the decimal character and grouping separator for numeric values.

B.1.17 NLS_SORT

Description This environment variable specifies the type of sort used for character data.

B.1.18 ORACLE_AFM

Description This environment variable specifies the location of AFM files. TK_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/guicommon/tk/admin/AFM

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.19 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:\orawin

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:\orawin

B.1.20 ORACLE_HPD

Description This environment variable specifies the location of HPD files. TK_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/guicommon/tk/admin/HPD

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.21 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.1.22 ORACLE_PPD

Description This environment variable specifies the location of PPD files. TK_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 PPD files in:

  ORACLE_HOME`/guicommon/tk/admin/PPD`

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.23 ORACLE_TFM

Description This environment variable specifies the location of TFM files. TK_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 of these variables, Oracle Reports looks for TFM files in

  ORACLE_HOME`/guicommon/tk/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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.24 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:\orawin\oin`

B.1.25 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`; that is, 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: Section 10.8.1, "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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.26 REMOTE

Description This environment variable specifies the default and remote SQL*Net driver to use when connecting through 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`*

Valid Values

- *netPrefix* Any valid SQL*Net driver prefix.
- *databaseName* The name of the local database.

Default Not defined.

Example `REMOTE=P:PIPER`

where

`P`: is the network prefix for Named Pipes

`PIPER` is the database name

B.1.27 REPORTS_ADD_HWMARGIN

Description (Windows only) This environment variable specifies whether to include the printer hardware-based left margin. By default, this margin is ignored. 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 reports independent of the printer hardware margin. These reports can then be deployed across various printers.

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 Note

- 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 Oracle Reports Builder have been disabled to ensure consistency with Oracle Reports Services.

B.1.28 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 not case-sensitive.

Valid Values ARABIC|HINDI|CONTEXT

Default ARABIC (Indo-Arabic)

Note: If you set the value of REPORTS_ENHANCED_BIDIHANDLING to YES, all the valid values of REPORTS_ARABIC_NUMERAL environment variable are supported.

See Also: Appendix B.1.47, "REPORTS_ENHANCED_BIDIHANDLING".

B.1.29 REPORTS_ALLOW_DB_CONNECT_STRING

Description This environment variable allows you to use DB connection strings in the userid parameter.

Valid Values Yes or No

Default No

Usage Notes

By default, DB connection strings are not allowed. However, tnsnames are allowed. You can specify the tnsnames alias as follows:

```
userid=user/pwd@server_tns_alias
```

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 Oracle Reports follows the Oracle BiDi algorithm.
- UNICODE Oracle Reports follows the **Unicode** BiDi algorithm.
- ENHANCED Oracle Reports follows the Enhanced BiDi algorithm.
- UNICODE_VARIANT Oracle Reports follows the XXX BiDi algorithm.

Default ORACLE

Note: If you set the value of REPORTS_ENHANCED_BIDIHANDLING is to YES, the following values are supported for the REPORTS_BIDI_ALGORITHM environment variable:

- ORACLE
- UNICODE
- UNICODE_VARIANT

The default value is UNICODE.

See Also: Appendix B.1.47, "REPORTS_ENHANCED_BIDIHANDLING"

For more information about the Unicode BiDi algorithm, refer to http://www.unicode.org/reports/tr9/.

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.

This environment variable is backward compatible.

Valid Values Any valid HTML attributes for the <BODY> tag.

Default Not defined.

Usage Notes

This environment variable is supported 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 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 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` key map file (see Section 18.14, "Using a Key Map File"), if map file configuration is used.

This environment variable is backward compatible.

Valid Values A valid path to the map file.

Default `$DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/cgicmd.dat`

Usage Notes

This environment variable is supported 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:\$DOMAIN_HOME\config\fmwconfig\servers\<WLS_SERVER_NAME>\applications\reports_<version>\configuration\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.

This environment variable is backward compatible.

Valid Values YES|NO

Default NO

Usage Notes

This environment variable is supported 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 components. You would typically add to this list when you want 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:** Oracle 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. The value of the REPORTS_CLASSPATH environment variable is platform dependent. For example, on a Windows platform the maximum value is 255 characters.

Valid Values The default values are mandatory. If any of the entries are removed, the Oracle Reports components 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/modules/oracle.jsp_11.1.1/ojsp.jar:$ORACLE_HOME/modules/oracle.javacache_11.1.1/cache.jar

Usage Notes

- The default value for the environment variable is required for Oracle Reports components 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 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.

B.1.37 REPORTS_CLIPBOARD_SIZE

Description It is used to specify the size of clipboard used in layout editor of reports builder copy/paste operation. Specify it to higher values if report has many objects in reports builder layout editor.

Values

The values of `REPORTS_CLIPBOARD_SIZE`:

1. 256KB clipboard, default
2. 1MB clipboard
3. 4MB clipboard

B.1.38 REPORTS_CONTAINSHTMLTAGS

Description This environment variable specifies whether Oracle Reports interprets the HTML formatting tags for all the supported output formats.

> **Note:** Oracle Reports' interpretation of inline HTML tags may be different from the browser's interpretation. As a result, a report designed with inline HTML tags in Oracle Reports 6*i*, Oracle9*i* Reports, or Oracle Reports 10*g* Release 1 (9.0.4) may generate a different HTML or HTMLCSS output in Oracle Reports 12c (12.2.1.3.0).

Valid Values

- `YES` Oracle Reports interprets the HTML tags for those objects whose Contains HTML Tags property is set to Yes.
- `NO` Oracle Reports does not interpret the HTML tags for the report, regardless of the object's Contains HTML Tags property setting. For HTML and HTMLCSS output, the browser will interpret the HTML formatting tags; for other output formats, the HTML tags themselves will appear as is in the report output. Set this environment variable to `NO` if you do not wish for Oracle Reports to interpret HTML formatting tags, and thus retain the behavior of prior releases.

  See Also: Section A.5.17, "CONTAINSHTMLTAGS" for more information on the implementation of inline HTML formatting tags.

Default YES

Usage Note

The command line keyword CONTAINSHTMLTAGS overrides the value of this environment variable.

B.1.39 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 through its own authentication mechanism (identical to the behavior of Oracle Reports 6*i*). Because the HTTP 1.0 protocol is stateless (that is, 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 (for example, 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 $DOMAIN_HOME/config/fmwconfig/servers/<WLS_SERVER_NAME>/applications/reports_<version>/configuration/cgicmd.dat %S forces users to enter their user name and password each time the report is called. See Section 18.14, "Using a Key Map File".

Valid Values Any number of minutes.

Default 30

Usage Note

This environment variable is supported 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.1.40 REPORTS_CUPS_PRINTING

Description This environment variable enables or disables sending the report in PDF format (instead of in postscript format) to the CUPS printer queue.

Valid Values

- `YES` Enables sending the report in PDF format to the CUPS printer queue.
- `NO` Disables sending the report in PDF format to the CUPS printer queue.

Default `NO`

Usage Note

After setting this environment variable to `YES`, you can use the following command-line parameters to send the report in PDF format to the printer queue:

```
DESFORMAT=pdf DESTYPE=printer DESNAME=cups_printer_queue_name
```

Note: *SRW.SET_PRINTER_TRAY()* built-in is ignored when the CUPS feature is used.

Example `REPORTS_CUPS_PRINTING=YES`

B.1.41 REPORTS_ENABLE_DBCLIENTID

Description This environment variable allows Reports to send the SSO ID as client ID to the database during login. This environment variable is backward compatible.

Valid Values

- `YES` Enables sending SSO ID as client ID in an SSO environment.
- `NO` Disables sending the SSO ID as client ID.

Default `No`

B.1.42 REPORTS_DB_AUTH

Description This environment variable specifies the database authentication template used to log on to the database. This environment variable is backward compatible.

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 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.1.43 REPORTS_DEFAULT_DISPLAY

Description This environment variable specifies whether to implement the following features:

- The elimination of the dependency on the `DISPLAY` variable (UNIX only)
- Using ScreenPrinter (`screenprinter.ppd`) for surface resolution for images and font information, which eliminates the dependency on having a valid printer defined (`PRINTER` and `TK_PRINTER` environment variables set to a valid printer, or a valid entry in `uiprint.txt`) for Reports Runtime (UNIX only).
- Advanced imaging support (all platforms)

  See Section 10.8.2, "Advanced Imaging Support".

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 the `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, HTML, HTMLCSS) and BMP format (for RTF output).
- On UNIX, `REPORTS_DEFAULT_DISPLAY=YES` overrides any value set for the `DISPLAY` variable. Even if the `DISPLAY` variable is defined, the X-Windows display surface will not be used by default. The surface resolution can be controlled with the entry in the `screenprinter.ppd`. For users upgrading from releases prior to Oracle Reports 10*g* Release 1 (9.0.4), this change may impact the appearance, number of pages, output file size, or performance of existing reports.
- To revert to the dependency on `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.
  - Set the `DISPLAY` variable to the active X-Windows display surface.
- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.44 REPORTS_DEFAULT_PIXEL_SIZE

Description This environment variable specifies a pixel size that overrides the display server's default pixel size when generating a report to HTML output. Normally, Oracle Reports takes its pixel size from the display server. If you are working with older reports that rely upon a pixel size that is different from that of the display server (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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.45 REPORTS_ENABLE_RTF_SPACING

This environment variable specifies whether to enable functionality that prevents truncation of multiline text in RTF output. This environment variable may not be needed in most circumstances, and should be set to `YES` only if you see truncation of multiline text in RTF output.

Valid Values

- `YES` Oracle Reports enables functionality to prevent truncation of multiline text in RTF output.
- `NO` Oracle Reports does not enable functionality to prevent truncation of multiline text in RTF output. Truncation may occur.

Default `NO`

B.1.46 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 `reports9i`

Usage Note

This environment variable is supported 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=oraclereports10g`

B.1.47 REPORTS_ENHANCED_BIDIHANDLING

Description This environment variable specifies whether or not to use the BIDI reshaping mechanism. You must set this environment variable to `YES` if you want to use the enhanced BiDi handling feature. If you do not set the value of the environment variable or you set the value of the environment variable to `NO`, Oracle Reports uses the old mechanism.

Valid Values

- `YES` Oracle Reports follows the new enhanced BiDi algorithm.
- `NO` Oracle Reports follows the old mechanism.

Default

`NO`

Usage Notes

If you set the value of `REPORTS_ENHANCED_BIDIHANDLING` to `YES`:

- All the valid values of `REPORTS_ARABIC_NUMERAL` environment variable are supported.
- The following values are supported for `REPORTS_BIDI_ALGORITHM` environment variable:
  - `UNICODE`
  - `ORACLE`
  - `UNICODE_VARIANT`
- The default value is `UNICODE`. If the `REPORTS_BIDI_ALGORITHM` environment variable is not specified and you set the `REPORTS_ENHANCED_BIDIHANDLING` to `YES`, then the default value is UNICODE.

Note: The `REPORTS_ENHANCED_BIDIHANDLING` variable indicates the mechanism used (old or new). The `REPORTS_BIDI_ALGORITHM` specifies which algorithm is used.

See Section B.1.30, "REPORTS_BIDI_ALGORITHM" and Section B.1.28, "REPORTS_ARABIC_NUMERAL".

B.1.48 REPORTS_ENHANCED_FONTHANDLING

Description (UNIX only) This environment variable specifies whether to use the font model, or revert to the Toolkit font handling mechanism of prior releases.

Valid Values

- YES Oracle Reports manages fonts using the font model.
- NO Oracle Reports manages fonts using the Toolkit font handling mechanism of prior releases.

Default YES

B.1.49 REPORTS_ENHANCED_SUBSET

Description This environment variable specifies whether to include the enhanced TTF font subsetting feature when generating a report. This environment variable is set to YES by default to ensure that the PDF file generated is accessible and searchable.

Valid Values YES|NO

Default YES

Usage Note

Oracle Reports uses the enhanced font subsetting implementation, by default. If you set REPORTS_ENHANCED_SUBSET=NO, Oracle Reports will to revert to the Type3 font subsetting implementation used in releases prior to Oracle Reports 10*g* Release 2 (10.1.2).

For more information on PDF font subsetting, refer to Section 11.1.2.2, "Font Subsetting".

B.1.50 REPORTS_FONT_DIRECTORY

Description (UNIX only) This environment variable specifies the location of all the font files for all TTF and TTC fonts used in your reports. It enables you to create reports that are easily portable across operating systems by eliminating the need to hard-code directory paths.

The directory specified by REPORTS_FONT_DIRECTORY is not given access by default. It is used only to search for files, and Oracle Reports handles any associated security settings separately if and when the target files are located.

Define REPORTS_FONT_DIRECTORY in the same fashion you define other environment variables on your operating system, keeping in mind platform-specific rules such as path length, and so on.

Valid Values Any directory on any drive.

Default $DOMAIN_HOME/reports/fonts

Usage Notes

- Only one directory may be specified.
- You cannot store a font file in the database.
- REPORTS_FONT_DIRECTORY is limited to 256 characters.
- The default folder $DOMAIN_HOME/reports/fonts is empty out-of-the-box.
- You must either save your font files in $DOMAIN_HOME/reports/fonts or set REPORTS_FONT_DIRECTORY to point the folder where the font files are saved.

B.1.51 REPORTS_GRAPH_IMAGE_DPI

Description This 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 `REPORTS_GRAPH_IMAGE_DPI=250`:

- The time taken to generate a report with a graph increases 5 to 6 times when compared to the time taken to generate the same report with the value set to 72 DPI.
- 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 want to change the JVM heap size value through REPORTS_JVM_OPTIONS to avoid the `Out Of Memory` error for the JVM.
- This environment variable is not supported in Oracle Reports distribution functionality, as it is specific to PDF and printer outputs only.

B.1.52 REPORTS_IGNORE_IMAGE_TAG_RES

Description This environment variable is useful when a report includes certain image formats that have the ability to store the physical size of the image, which usually includes resolution and pixel dimensions. To ensure the image is not scaled to the physical dimensions, you can set this environment variable to `YES` to specify that Oracle Reports should ignore image resolution information and only use the pixel dimensions of the image. This ensures that this type of image from a database column is displayed correctly instead of displaying as a thumbnail.

Valid Values

- `YES` Oracle Reports ignores image resolution information, and uses only the pixel dimensions of the image.
- `NO` Oracle Reports does not ignore the image resolution information in the image.

Default `NO`

B.1.53 REPORTS_IGNORE_SET_ROLE_ERROR

Description

Valid Values

- YES
- NO

Default NO

B.1.54 REPORTS_JPEG_QUALITY_FACTOR

Description This environment variable specifies the level of image quality desired for JPEG images. It provides control over the trade-off between JPEG image quality and size of the image. The better the quality of the image, the greater the image file size.

Valid Values 0 through 100

Default 100 (highest quality)

Usage Notes

- On Windows, use the registry to specify the value. On Unix/Linux, set the environment variable in reports.sh.
- If REPORTS_JPEG_QUALITY_FACTOR is not specified or incorrectly specified (for example, set to a string or an out of range value), the default value is used.
- A value of 75 provide a good quality image, while ensuring a good compression ratio.

B.1.55 REPORTS_JVM_OPTIONS

Description This environment variable specifies any JVM options that you want Oracle Reports Builder, Reports Runtime, or Reports Converter 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, generate random or non-sequential job IDs, 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 MB to avoid the Out Of Memory error when running reports with large graphs or running big reports.
- When the Reports Engine starts up, it checks for JVM options specified in the *rwserver*.conf file in the jvmoptions attribute of the engine element. See Section 7.2.1.8, "engine". If specified, the JVM options set in *rwserver*.conf override the value of the REPORTS_JVM_OPTIONS environment variable. If not specified in *rwserver*.conf, Oracle Reports uses the JVM options specified by the REPORTS_JVM_OPTIONS environment variable.
- When running reports with Reports Builder, Reports Runtime, and Reports Converter, JVM options specified on the command line with the JVMOPTIONS command line keyword override JVM options specified by the REPORTS_JVM_OPTIONS environment variable.

- To generate random and non-sequential job IDs, you can set REPORTS_JVM_OPTIONS to -Djobid=random and export REPORTS_JVM_OPTIONS.

B.1.56 REPORTS_NETWORK_CONFIG

Description This environment variable should be set only if you want rwclient, rwrqm, rwcgi, or Oracle Forms Services to use a custom network configuration file. If this environment variable is not set, then these components will use the default network configuration file (rwnetwork.conf). For more information about rwnetwork.conf, see Section 7.5, "Network Configuration File".

Valid Values A valid custom network configuration file in ${DOMAIN_HOME}/config/fmwconfig/components/ReportsServerComponent/<reports_server_name>/rwnetwork.conf

Default rwnetwork.conf

B.1.57 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 necessary when the required character set mapping for NLS_LANG to XML IANA-defined 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 (for example, JA16SJIS) to what is expected in the XML specification (for example, 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.

Example

WISO-8859-8=ISO-8859-8-1;CSEUCKR=EUC-KR;WINDOWS-949=EUC-KR;EUC-CN=GBK;WINDOWS-936=GBK

B.1.58 REPORTS_NO_DUMMY_PRINTER

Description This environment variable, together with other printer and display environment variables and settings, 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.

  If the uiprint.txt file does not contain a valid entry (that is, 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:** Section 10.8.1, "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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.59 REPORTS_NO_HTML_SPACE_REPLACE

Description

This environment variable specifies whether spaces should not be replaced with   in HTML or HTMLCSS output.

Oracle Reports maps HTML metadata characters in the data retrieved for a field to the appropriate encoding. That is, Oracle Reports automatically maps:

- < to <
- " 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 preceding example are used to show preceding spaces; they are not part of the sentence.

B.1.60 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 and spreadsheet 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 Section 10.8.2, "Advanced Imaging Support" for more information.

B.1.61 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 eliminating the need to hard-code directory paths.

The directories specified by REPORTS_PATH are not given access by default. They are used only to search for files, and Oracle Reports handles any associated security settings separately if and when the target files are located.

In addition to directory paths, you can specify the keyword DB when you define REPORTS_PATH. This instructs Oracle Reports 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
```

Oracle Reports will first search the directory `/home/tkostin/pay`. If it cannot find the file in question, it will search `/home/tkostin/receive`.

Valid Values Any directory on any drive.

Default `%ORACLE_HOME%\REPORT\DEMO; %ORACLE_HOME%\REPORT\DEMO\BITMAP; %ORACLE_HOME%\REPORT\DEMO\REQFILES`

Usage Note

- Define `REPORTS_PATH` as you would define other environment variables on your operating system, considering platform-specific rules such as for the path length and so on.

  The permissible length of the `REPORTS_PATH` value depends on the operating system. All operating systems support up to 256 characters; but some might support a longer value.

- If you specify a path for the `sourceDir` attribute of the `engine` element in the Reports Server configuration file (`rwserver.conf`), the `sourceDir` value will override the values you set here.

Example

```
REPORTS_PATH=C:\oracle\apps\reports;C:\myfiles
```

B.1.62 REPORTS_RESTRICT_DIRECTORIES

This environment variable specifies whether read/write access to directories is restricted. If set to `YES`, Oracle Reports has read/write access only to directories specified by `REPORTS_PATH`.

Valid Values `YES|NO`

Default `NO`

B.1.63 REPORTS_RESOURCE

This environment variable specifies the location of the resource files required for reports. This path must include the globalization support directory extension when specifying the location of the resource files.

Valid Values Any directory on any drive.

Default `%ORACLE_HOME%\reports\res`

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.1.64 REPORTS_SERVER

Description This environment variable specifies the default Reports Server for Web Cartridge or Web CGI requests. When this environment variable is set, you can omit the `SERVER` command line keyword in report requests to process them using the default Reports Server, or you can include the `SERVER` command line keyword to override the default.

This environment variable is backward compatible.

Valid Values Any Reports Server service entry name.

Usage Note

This environment variable is supported 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. The default Reports Server name is defined in the Oracle Reports Servlet (`rwservlet`) configuration file (`rwservlet.properties`), as described in Section 7.3.1.1.1, "server".

Example `REPORTS_SERVER=my_rep_server`

B.1.65 REPORTS_SOLARIS_9

Description This environment variable resolves a synchronization issue between native Motif libraries and JDK UI packages on Solaris 2.9. When `REPORTS_SOLARIS_9=YES`, Reports Builder responds as expected. If you set `REPORTS_SOLARIS_9=NO` in a Solaris 2.9 environment, Reports Builder may stop responding when invoking either the Report Wizard or Data Wizard.

Valid Values `YES|NO`

Default `YES` on Solaris 2.9; `NO` on other platforms.

B.1.66 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.1.67 REPORTS_SRWRUN_TO_SERVER

Description

This environment variable specifies whether to allow the SERVER or USERID keywords when running a report using the SRW.RUN_REPORT built-in procedure.

Valid Values YES|not set

Default not set

Usage Notes

- Keywords SERVER and USERID with SRW.RUN_REPORT are deprecated. If you have reports created in prior releases that use these keywords with SRW.RUN_REPORT, you can set REPORTS_SRWRUN_TO_SERVER=YES to continue to run these reports with the current release.
- You may encounter issues when attempting to run reports created in prior releases asynchronously. For this reason, it is important to migrate your reports to the latest Oracle Reports release as soon as possible.

> **Note:** For a description of the SRW built-in package, including the SRW.RUN_REPORT built-in procedure, see the *Oracle Reports online Help*.

B.1.68 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 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.1.69 REPORTS_STDIN_PASSWORD

Description Passing userid as an argument when using the Reports converter (rwconverter) may lead to security risks. In addition to the interactive dialog mode and command line mode already available, the converter can now accept the connect string via standard input. To pass the userid in a secure mode, perform the following: Set the environment variable REPORTS_STDIN_PASSWORD to YES.

Run the reports converter without any connect string. Enter the connect string after the converter has started. Run the reports converter using redirection to pass the password

to the converter. (This is especially useful in compiling several Reports in a script.) For example:

```
#!/bin/sh
 echo "Enter userid"
 read -s myuserid
 for i in 'ls *.rdf'
 do
 echo Compiling Report $i ....
 rwconverter.sh command_line_arguments> <<<
     "$myuserid"
 done
```

B.1.70 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 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.1.71 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 Oracle Reports. The "prefix" attribute can be changed to avoid naming conflicts independent of the "uri" attribute.

B.1.72 REPORTS_TMP

Description This environment variable specifies the directory in which you wish to store Oracle Reports Builder temporary files. Oracle 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.

It is recommended that you check the TMP/TEMP values while installing *FMW 11g/11gR2*. Since REPORTS_TMP can inherit these variable values, it is better to set them to a reliable directory, e.g. C:\TEMP before proceeding for the installation.

Unreliable subdirectories are the numbered subdirectories, e.g. *C:\Users\xxxxx\AppData\Local\Temp\3*, that are by default created by Microsoft (mstsc). Although these directories are used in TMP/TEMP environment variables, at some point of time they can be deleted by the Operating System.

Valid Values Any directory on any drive.

Default Not defined.

Example `REPORTS_TMP=C:\tmp`

B.1.73 REPORTS_USEREXITS

Description This environment variable specifies the libraries for use by Oracle Reports. These libraries are program modules created by you to be called by Oracle Reports.

`REPORTS_USEREXITS` can specify multiple libraries. On Windows, use a backslash (`\`) to separate directories in a path, and a semicolon (;) to separate complete paths. On UNIX, use a forward slash (`/`) to separate directories in a path, and a colon (:) to separate complete paths.

If this value is not explicitly set, Oracle Reports looks for `rwxtb.dll` according to the path variable of the system.

Note: You can call Java methods using the `ORA_JAVA` built-in package and the Java Importer. This reduces the need to have user exits in a report and allows for a more open and portable deployment. You may also use the `ORA_FFI` built-in package, which provides a foreign function interface for invoking C functions in a dynamic library. With the availability of these built-in packages, the use of user exits is deprecated in Oracle Reports, though makefiles are still be supplied to permit you to continue to work with existing user exits.

For backward compatibility, the prior name `REPORTS_USEREXIT` is allowed for this environment variable.

Valid Values Any user exit library (along with its absolute path).

Default Not defined.

Example

On Windows:

```
REPORTS_USEREXITS=C:\mydll.dll;d:\mynew.dll;e:\bin\speed.dll
```

On UNIX:

```
REPORTS_USEREXITS=/usr/oracle/mylib.so:/usr/oracle/myfolder/speed.so
```

B.1.74 REPORTS_UTF8_XMLOUTPUT

Description This environment variable specifies whether the `UTF8` character set is used instead of the `NLS_LANG` character set. This environment variable is in effect only when the encoding attribute is *not* specified by the XML Prolog Value property (see the *Oracle Reports online Help* for a description of the XML Prolog Value property).

Valid Values

`YES` Assigns the `UTF8` character set (when the XML Prolog Value property is not set).

`NO` Assigns the `NLS_LANG` (or IANA-defined) character set (when the XML Prolog Value property is not set).

Default `YES`

B.1.75 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.1.76 TK_PRINT

Description (UNIX only) This environment variable specifies the print command to be executed on UNIX for Oracle Reports 6*i*. In later releases, `TK_PRINT` is *obsolete*; you can achieve the same results by using the printing script file: `$DOMAIN_HOME/reports/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`

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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.77 TK_PRINT_STATUS

Description (UNIX only) This environment variable specifies the command executed to validate the printer. To ensure that 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: Section 10.8.1, "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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.78 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; that is, 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: Section 10.8.1, "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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.79 TK_AFM

Description This environment variable specifies the location of AFM files. `TK_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`/guicommon/tk/admin/AFM`

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.80 TK_HPD

Description This environment variable specifies the location of HPD files. `TK_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/guicommon/tk/admin/HPD

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.81 TK_PPD

Description This environment variable specifies the location of PPD files. TK_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 PPD files in:

  $DOMAIN_HOME/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/guicommon/tk/admin/PPD

- Printing on UNIX requires some setup and configuration to create the proper printing environment. For detailed information, refer to Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.82 TK_TFM

Description This environment variable specifies the location of TFM files. TK_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 of these variables, Oracle Reports looks for TFM files in

  ORACLE_HOME/guicommon/tk/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 Chapter 10, "Printing on UNIX with Oracle Reports".

B.1.83 TNS_ADMIN

Description

This environment variable points to the directory location containing tnsnames.ora which has the DB connection details.

Valid Values

Any directory or any drive which has the file `tnsnames.ora`

Default Value

By default, the value of the environment variable `TNS_ADMIN` will be set to `$DOMAIN_HOME/config/fmwconfig/` for both in-process server and standalone servers.

Usage Notes

If you want to change the value of `TNS_ADMIN`:

- For Standalone server: Edit the reports_process.xml file which is located at `$DOMAIN_HOME/system_components/ReportsServerComponent/<reports_server_name>/data/nodemanager/`
- For In-process server: Edit the `setStartupEnv.sh` file which is located at `$DOMAIN_HOME/bin`

B.1.84 USERNAME

Description This environment variable specifies the 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.1.85 USER_NLS_LANG

Description This environment variable specifies the language for the Oracle Reports Runtime component. Chapter 23, "Implementing Globalization and Bidirectional Support" contains additional detailed information about this environment variable, including a table of valid values.

C

Batch Registering Reports in Oracle Portal

If you have a number of reports that you wish to register in Oracle Portal, it is often preferable to register them as a group in a batch script rather than individually in the Oracle 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 Oracle Portal, 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 Keywords" 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` keywords. To create a functional package in Oracle Portal, you must specify the `P_SERVERS`, `P_PRIVILEGE`, `P_TYPES`, `P_FORMATS` keywords in addition to the keywords 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;"
```

This command line generates 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 on the contents of this SQL script file, refer to Section C.3, "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 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 using the `P_PARAMETERS` keyword. 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 Oracle Portal, you must run the script generated for you by rwconverter:

1. Start SQL*Plus and connect to the Oracle 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 Oracle Portal for each report listed in the script with the specified parameters.
3. Log in to Oracle 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 Oracle Portal at once, do the following:

1. In a text editor, create a SQL script file (for example, 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 Oracle Portal.

2. Start SQL*Plus and log in to the Oracle 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 Oracle 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 Oracle 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 Fusion Middleware 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 Fusion Middleware 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 Oracle Portal. P_NAME corresponds to the Name field in the Create Report Definition File Access wizard. For example: `P_NAME=>'Earnings'`
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 Oracle Portal. For more information, refer to Chapter 16, "Deploying Reports in Oracle 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 (Cont.) 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 Oracle Portal. For more information on creating an availability calendar, see Chapter 16, "Deploying Reports in Oracle Portal".
P_TYPES	Is the destination types to which the report definition file defined in the P_FILENAME parameter can be sent (for example, 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 (for example, 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' **Note**: If the destination format for the report is DELIMITEDDATA, it may not be possible to batch register the report. As a workaround, you can define a different destination format, then batch register the report, and later manually edit the report to DESFORMAT=DELIMITEDDATA.
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 Oracle Portal. For more information, refer to Chapter 16, "Deploying Reports in Oracle 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 (Cont.) 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.

P_TRIGGER	Is the validation trigger written in PL/SQL that returns a boolean statement (for example, 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;'

D

Troubleshooting Oracle Reports Services

This appendix describes common problems that you might encounter when deploying your reports using Oracle Reports Services and explains how to solve them. It also gives detailed instructions on how to diagnose problems. It contains the following topics:

D.1 Problems and Solutions

This section describes common problems and solutions. It contains the following topics:

D.1.1 Hanging Report Requests

When running report requests with Reports Server, the report request may "hang" for various reasons. This can lead to stability issues if not noticed in time. This section highlights such scenarios, explains the issues, how you can identify such patterns, take corrective measures, and gather sufficient information to raise such issues with Oracle Support Services.

To begin with, it is important to understand how Reports Server identifies duplicate jobs. When a job is submitted to Reports Server, it checks whether a similar job exists in its job queue. If it finds a currently running job that is the same as the submitted job, then Reports Server considers the submitted job a duplicate job and the currently running job as the master job. Reports Server does not execute the duplicate job; instead, it waits for the master job to finish and passes the same output to the duplicate job. Although an idle engine is available, the duplicate job is not submitted to the engine. This is expected behavior and does not mean that the request is hanging.

In addition to the Solutions provided in this section, refer to Section 24.3, "Tuning Reports Server Configuration".

Note: Scalability improvements in Oracle Reports 11.1.2/11.1.1 and 12*c* Release (12.2.1.3) improve the stability of Reports Server to ensure report requests complete successfully.

Problem 1

Master job "hangs" before finishing.

Solution 1

If a master job hangs for some reason, then the next duplicate job in line is made the master job.

Check the `engineResponseTimeOut` attribute in the `engine` element of the *`rwserver`*`.conf` file (see Section 7.2.1.8, "engine"). Set this attribute judiciously to avoid server instability. This enables Reports Server to automatically detect and recover from this type of hanging situation. You can also use the `showjobs` command to end the hanging job and allow Reports Server to continue processing other requests. For information about the `showjobs` command, see Section A.8.8, "SHOWJOBS".

For example, consider a scenario where you have a set of reports. The largest report takes a maximum of 5 minutes to run. In this case, you can set `engineResponseTimeOut` to 5 minutes.

Notes:

When an engine is executing the job, the engine updates the server with the latest status, such as formatting page 1, 2, and so on. If Reports Server does not receive any update from the engine for more than 5 minutes, it is assumed that the engine is hanging and therefore, Reports Server stops the engine.

When you have reports of various complexities that take 1 minute to 1 hour to run, you should specify `ENGINERESPONSETIMOUT` on the command line while running the report (see Section A.6.4, "ENGINERESPONSETIMEOUT").

If you have interactive jobs as well as scheduled and batch jobs, it is good practice to start one server for interactive jobs and one for batch and scheduled jobs. For performance and stability reasons, you should avoid using the same server for both interactive and batch/scheduled jobs.

Despite setting the `engineResponseTimeOut` attribute (or `ENGINERESPONSETIMEOUT` keyword on the command line) judiciously, if you still encounter instability and crashes, perform the following steps to report the problem to Oracle Support Services:

1. Enable server tracing and logging (see Section 24.2.5, "Tracing Report Execution"). If it is not possible to enable tracing, enable logging alone by setting the `log` element's `option` attribute to `failedJobs` in the *`rwserver`*`.conf` file (see Section 7.2.1.12, "log"). When you enable logging, you can see the failed job reports in the `reports.log` file. Identify the report that is failing or causing the engine to hang.
2. Enable engine diagnostic logging by modifying the `engine` element to include the `diagnosis` property in the *`rwserver`*`.conf` file (see "Properties" in Section 7.2.1.8, "engine"), then run the report that you identified in Step 1 to reproduce the hang.
3. Report the hang to Oracle Support Services with the following information:
   - *`rwserver`*`.conf` file.
   - `reports.log` file.
   - Engine diagnostic output when the hang is reproduced.
   - Report definition file so that Oracle Support Services can reproduce the problem.

Problem 2

Reports Server stops responding or crashes when running report requests, exhibited by any of the following:

- When a job is submitted through a browser, the browser seems to hang (no response).
- A job is not submitted to an engine although the engine is idle.
- Web commands do not work and the browser times out after some time.
- Scheduled jobs are not run.

Solution 2

Restart Reports Server to attempt to recover from this problem. If the problem persists, report it to Oracle Support Services with the following information:

- *`rwserver`*`.conf` file.
- Approximate load on Reports Server at the time of the hang.
- Thread dump of Reports Server, which you can obtain as follows:
  - On Solaris, use the `kill -3` *`server_pid`* command when Reports Server hangs. This command writes the thread information to the console output. To redirect the thread information and error streams from the console to a file, modify the `rwserver.sh` file in the `$DOMAIN_HOME/reports/bin/` directory. For example:

    ```
    $DOMAIN_HOME/reports/bin/rwserver
    ```

    Note: This example is for the UNIX k shell. The code may be slightly different if you are using some other shell.

If you are using the in-process Reports Server, use the `Give kill -3` to `WLS_REPORTS` managed server process. The thread dump will be in `FMW_HOME/user_projects/domains/domain_name/servers/WLS_REPORTS/logs/WLS_REPORTS.out`

- On Windows, the `kill -3` command does not work. Instead, at a command prompt, type the command specified in Table D–1 to start Reports Server; when the issue is reproduced, shift focus to the command prompt window, then press `Control+Break` to get the thread dump.

Table D–1 Commands to Obtain Thread Dump on Windows

Reports Server Version	Command
11.1.1.x.x	Set the environment variable `COMPONENT_CONFIG_PATH` to the Reports Server configuration directory and run the following command: *ORACLE_HOME*`/jdk/bin/java -Xmx256M -classpath %REPORTS_CLASSPATH% oracle.reports.server.RWServer oracle_home=`*ORACLE_HOME* `server=`*server_name* `showui=yes|nobatch=yes|no`
10.1.2.0.2	*ORACLE_HOME*`/jdk/bin/java -Xmx256M -classpath %REPORTS_CLASSPATH% oracle.reports.server.RWServer oracle_home=`*ORACLE_HOME* `server=`*server_name* `showui=yes|no batch=yes|no`
9.0.4	*ORACLE_HOME*`/jdk/bin/java -Xbootclasspath/p:$OH/vbroker4/lib/vbjboot.jar -Xmx256M -classpath %REPORTS_CLASSPATH% oracle.reports.server.RWServer oracle_home=`*ORACLE_HOME* `server=`*server_name* `showui=yes|no batch=yes|no`
9.0.2	*ORACLE_HOME*`/jdk/bin/java -Xmx256M -classpath %REPORTS_CLASSPATH% oracle.reports.server.RWServer oracle_home=`*ORACLE_HOME* `server=`*server_name* `showui=yes|no batch=yes|no`

Problem 3

The in-process Reports Server fails to start and the browser displays the following message while trying to run a report with the in-process Reports Server:

```
REP-52266:  The in-process Reports Server failed to start.
```

When the standalone server is started, it shuts down immediately.

Solution 3

Enable tracing (see Section 24.2.5, "Tracing Report Execution") and start the in-process Reports Server. The default Reports Server `rwserver_diagnostic.log` file should capture the actual cause of the problem:

- **Reports Server has failed to initialize one of the pluggable data sources or destinations**. Correct the configuration for the pluggable data source (PDS) or destination and restart Reports Server. For general information about PDSs, see the **Pluggable Data Sources** section of the *Oracle Reports online Help*.
- **The engine has failed to start**. Check the `rwEng_{engNo}_diagnostic.log` file in the *`$DOMAIN_HOME/servers/WLS_REPORTS/logs/reports`* directory. This file must contain the following lines:

```
Debug 50103 (EngineImpl:EngineImpl): CInitEngine returns 0
Info 55003 (RWEngine:init): Register this engine to Oracle Reports Server
```

server_name

If the rwEng_{engNo}_diagnostic.log file does not contain these lines, it means that the engine has failed to start.

If the CinitEngine return value in the file is negative, then it represents an error in initializing the Reports Engine.

If the CinitEngine return value is not equal to zero, check the System environment variable PATH if you are using Windows and the LD_LIBRARY_PATH environment variable in setdomainenv.sh file which is located in DOMAIN_HOME/bin if you are using Solaris. For the in-process Reports Server, the values of PATH and LD_LIBRARY_PATH are taken from the System environment variable PATH for windows and LD_LIBRARY_PATH in setdomainenv.sh file which is located in DOMAIN_HOME/bin for Solaris.

Problem 4

Reports Engine crashes or hangs when running report requests.

Solution 4

Case 1: Consider the scenario where Reports Server is running thousands of reports every day, printing reports, and publishing them to the Web. In this scenario, the browser may wait for the response and eventually time out. Even Web commands to see the job queue may not work.

Turn on tracing (see Section 24.2.5, "Tracing Report Execution") and when this problem occurs, take a thread dump by running the `kill -3 server_pid` command on Solaris (as described under **Solution 2**. The following lines of code are the result of running the `kill -3 server_pid` command. These lines indicate a hang when Reports Server is trying to write the report to a network drive:

```
"RequestProcessor[7]" daemon prio=5 tid=0x1835f210 nid=0x181c waiting on condition
[224cf000..224cfd88]
   at java.io.FileOutputStream.write (Native Code)
   at oracle.reports.utility.copyFile (Utility.java:424)
   at oracle.reports.server.DesFile.sendFile(DesFile.java:74)
   at oracle.reports.server.Destination.send(Destination.java:484)
   at oracle.reports.server.JobObject.distribute(JobObject.java:1582)
   at oracle.reports.server.JobManager.updateJobStatus(JobManager.java:2231)
   at oracle.reports.server.EngineCommImpl.updateEngineJobStatus(
      EngineCommImpl.java:134)
   at oracle.reports.server._EngineCommImplBase._invoke(
      _EngineCommImplBase.java:94)
   at com.sun.corba.se.internal.corba.ServerDelegate.dispatch
      (ServerDelegate.java:353)
   at com.sun.corba.se.internal.iiop.ORB.process(ORB.java:280)
   at com.sun.corba.se.internal.iiop.RequestProcessor.process
      (RequestProcessor.java:81)
   at com.sun.corba.se.internal.orbutil.ThreadPool$PooledThread.run
      (ThreadPool.java:106)
```

The trace file for this scenario is as follows:

```
[2005/5/31 6:26:47:321] Info 50132 (JobObject:reset): jobid = 15 Get command line:
server=vin report=c:\backup\reps\emp.rdf destype=file desformat=html
desname=c:\test.html userid=scott@ora9i authid=vnhegde
[2005/5/31 6:26:48:92] Debug 50103 (JobManager:firstToRun): job 15 is first to run
[2005/5/31 6:26:48:212] Debug 50103 (ConnectionImpl:runJob): Job queue for jobid =
  15 is 0
```

```
[2005/5/31 6:26:48:212] Debug 50103 (ConnectionImpl:runJob): jobid = 15 is in
  current queue
[2005/5/31 6:26:48:212] Debug 50103 (ConnectionImpl:runJob): Calling
  findDuplicatedJob for jobid = 15
[2005/5/31 6:26:48:212] Debug 50103 (JobManager:findDuplicatedJob): Found no
  duplicated job for job 15
[2005/5/31 6:26:48:212] Debug 50103 (ConnectionImpl:runJob): No Duplicate jobs for
  jobid = 15
[2005/5/31 6:26:48:212] Debug 50103 (ConnectionImpl:runJob): Job 15 is Enqueued
[2005/5/31 6:26:48:212] Debug 50103 (JobManager:firstToRun): job 15 is first to
  run
[2005/5/31 6:26:48:212] Debug 50103 (JobManager.runJobLocal): Trying to get engine
  for Job 15
[2005/5/31 6:26:48:212] Debug 50103 (EngineManager:getIdleEngine): Target max
  engines = 1
[2005/5/31 6:26:48:222] Debug 50103 (EngineManager:getIdleEngine): rwEng-0 is used
  = true
[2005/5/31 6:26:48:222] Debug 50103 (EngineManager:getIdleEngine): rwEng-0 state
  is 1
[2005/5/31 6:26:48:222] State 56004 (EngineInfo:setState): Engine rwEng-0 state
  is: Reserved
[2005/5/31 6:26:48:222] Debug 50103 (JobManager.runJobLocal): Job 15 got Engine
  rwEng-0
[2005/5/31 6:26:48:222] Debug 50103 (JobManager:runJobInEngine): Job 15 calling
  setCommand on engine rwEng-0
[2005/5/31 6:26:48:222] Debug 50103 (EngineManager:updateEngineState): Engine
  rwEng-0 status is 3
[2005/5/31 6:26:48:222] State 56004 (EngineInfo:setState): Engine rwEng-0 state
  is: Running
[2005/5/31 6:26:48:222] Debug 50103 (EngineManager:updateEngineState): Engine
  rwEng-0 status is 5
[2005/5/31 6:26:48:222] State 56004 (EngineInfo:setState): Engine rwEng-0 state
  is: Idle
[2005/5/31 6:26:48:232] Debug 50103 (JobManager:runJobInEngine): Send job 15 to
  engine rwEng-0
[2005/5/31 6:26:48:232] Debug 50103 (EngineManager:updateEngineState): Engine
  rwEng-0 status is 3
[2005/5/31 6:26:48:232] State 56004 (EngineInfo:setState): Engine rwEng-0 state
  is: Running
[2005/5/31 6:26:48:482] State 56016 (JobManager:updateJobStatus): Job 15 status
  is: Running the report Initializing report
[2005/5/31 6:26:48:482] Debug 50103 (JobManager:updateJobStatus): Finished
  updating job: 15
[2005/5/31 6:26:50:856] State 56016 (JobManager:updateJobStatus): Job 15 status
  is: Running the report Formatting page 1
[2005/5/31 6:26:50:856] Debug 50103 (JobManager:updateJobStatus): Finished
  updating job: 15
[2005/5/31 6:26:52:468] Debug 50103 (RWCacheItem:addFile): add file
  'test33347112.htm' for job 15
[2005/5/31 6:26:52:468] Debug 50103 (RWCache:updateCurrentCapacity): Current cache
  capacity is 197239
```

In this trace file, note the following:

- A job with ID `15` is submitted at `6:26:47:321`
- A duplicate job is checked for at `6:26:48:212`
- `rwEng-0` is obtained at `6:26:48:222`
- The engine started running at `6:26:48:222`

- The first page is formatted at `6:26:50:856`

After this there is no update on the job. The `Finished successfully` line is not present. This indicates that there is a problem with the job.

The following example shows a trace file for a job that finished successfully:

```
[2005/5/31 6:25:57:198] Info 50132 (JobObject:reset): jobid = 14 Get command line:
server=vin report=c:\backup\reps\emp.rdf destype=file desformat=html
desname=c:\test.html userid=scott@ora9i authid=vphegde
[2005/5/31 6:25:58:80] Debug 50103 (ConnectionImpl:runJob): Job queue for jobid =
  14 is 0
[2005/5/31 6:25:58:90] Debug 50103 (ConnectionImpl:runJob): jobid = 14 is in
  current queue
[2005/5/31 6:25:58:90] Debug 50103 (ConnectionImpl:runJob): Calling
  findDuplicatedJob for jobid = 14
[2005/5/31 6:25:58:90] Debug 50103 (JobManager:findDuplicatedJob): Found no
  duplicated job for job 14
[2005/5/31 6:25:58:90] Debug 50103 (ConnectionImpl:runJob): No Duplicate jobs for
  jobid = 14
[2005/5/31 6:25:58:90] Debug 50103 (ConnectionImpl:runJob): Job 14 is Enqueued
[2005/5/31 6:25:58:90] Debug 50103 (JobManager:firstToRun): job 14 is first to run
[2005/5/31 6:25:58:90] Debug 50103 (JobManager.runJobLocal): Trying to get engine
  for Job 14
[2005/5/31 6:25:58:90] Debug 50103 (EngineManager:getIdleEngine): Target max
  engines = 1
[2005/5/31 6:25:58:90] Debug 50103 (EngineManager:getIdleEngine): rwEng-0 is used
  = true
[2005/5/31 6:25:58:90] Debug 50103 (EngineManager:getIdleEngine): rwEng-0 state is
  1
[2005/5/31 6:25:58:90] State 56004 (EngineInfo:setState): Engine rwEng-0 state is:
  Reserved
[2005/5/31 6:25:58:90] Debug 50103 (JobManager.runJobLocal): Job 14 got Engine
  rwEng-0
[2005/5/31 6:25:58:90] Debug 50103 (JobManager:runJobInEngine): Job 14 calling
  setCommand on engine rwEng-0
[2005/5/31 6:25:58:100] Debug 50103 (EngineManager:updateEngineState): Engine
  rwEng-0 status is 3
[2005/5/31 6:25:58:100] State 56004 (EngineInfo:setState): Engine rwEng-0 state
  is: Running
[2005/5/31 6:25:58:100] Debug 50103 (EngineManager:updateEngineState): Engine
  rwEng-0 status is 5
[2005/5/31 6:25:58:100] State 56004 (EngineInfo:setState): Engine rwEng-0 state
  is: Idle
[2005/5/31 6:25:58:100] Debug 50103 (JobManager:runJobInEngine): Send job 14 to
  engine rwEng-0
[2005/5/31 6:25:58:110] Debug 50103 (EngineManager:updateEngineState): Engine
  rwEng-0 status is 3
[2005/5/31 6:25:58:110] State 56004 (EngineInfo:setState): Engine rwEng-0 state
  is: Running
[2005/5/31 6:25:58:350] State 56016 (JobManager:updateJobStatus): Job 14 status
  is: Running the report Initializing report
[2005/5/31 6:25:58:350] Debug 50103 (JobManager:updateJobStatus): Finished
  updating job: 14
[2005/5/31 6:26:0:663] State 56016 (JobManager:updateJobStatus): Job 14 status is:
  Running the report Formatting page 1
[2005/5/31 6:26:0:663] Debug 50103 (JobManager:updateJobStatus): Finished updating
  job: 14
[2005/5/31 6:26:2:256] Debug 50103 (RWCacheItem:addFile): add file
  'test54106877.htm' for job 14
[2005/5/31 6:26:2:256] Debug 50103 (RWCache:updateCurrentCapacity): Current cache
```

```
  capacity is 182329
[2005/5/31 6:26:2:286] State 56016 (JobManager:updateJobStatus): Job 14 status is:
  Finished successfully
[2005/5/31 6:26:3:7] Debug 50103 (JobManager:notifyWaitingJobs): Master job 14
  notify its duplicated jobs.
[2005/5/31 6:26:3:7] Debug 50103 (JobManager:updateJobStatus): Finished updating
  job: 14
[2005/5/31 6:26:3:7] Debug 50103 (EngineManager:updateEngineState): Engine rwEng-0
  status is 1
[2005/5/31 6:26:3:7] State 56004 (EngineInfo:setState): Engine rwEng-0 state is:
  Ready
[2005/5/31 6:26:3:57] Info 56013 (ConnectionManager:release): Connection 1 is
  released
```

In this trace file, after formatting the page 1, note the following:

- The job finished successfully at `6:26:2:286`
- Duplicate jobs are notified at `6:26:3:7`
- Connection is released at `6:26:3:57`

These lines were not present in the first example. All jobs must contain these lines in the Reports Server trace files. A missing event or abrupt end means that the job has not finished successfully and is a potential cause for the hang.

Case 2: Consider the scenario where the following error displays:

```
REP-56048: Engine rwEng-0 crashed, job Id: 17
```

In this scenario, check the Reports Server and engine trace files. A typical crash resembles the following in the Reports Server trace file:

```
[2005/6/1 3:38:35:156] Exception 50125 (org.omg.CORBA.COMM_FAILURE:   vmcid: SUN
  minor code: 208 completed: Maybe
    at com.sun.corba.se.internal.iiop.IIOPConnection.purge_
    calls(IIOPConnection.java:438)
    at com.sun.corba.se.internal.iiop.ReaderThread.run(ReaderThread.java:70)
): Internal error org.omg.CORBA.COMM_FAILURE:   vmcid: SUN  minor code: 208
  completed: Maybe
[2005/6/1 3:38:35:156] Info 56029 (EngineManager:shutdownEngine): Shutting down
  engine rwEng-0
[2005/6/1 3:38:36:137] Exception 50125 (org.omg.CORBA.COMM_FAILURE:   vmcid: SUN
  minor code: 201  completed: No
    at com.sun.corba.se.internal.iiop.ConnectionTable.getConnection
    (ConnectionTable.java:148)
    at com.sun.corba.se.internal.iiop.ConnectionTable.getConnection
      (ConnectionTable.java:65)
    at com.sun.corba.se.internal.iiop.GIOPImpl.getConnection(GIOPImpl.java:67)
    at com.sun.corba.se.internal.corba.ClientDelegate.createRequest
      (ClientDelegate.java:652)
    at com.sun.corba.se.internal.corba.ClientDelegate.createRequest
      (ClientDelegate.java:594)
    at com.sun.corba.se.internal.corba.ClientDelegate.request
      (ClientDelegate.java:886)
    at org.omg.CORBA.portable.ObjectImpl._request(ObjectImpl.java:431)
    at oracle.reports.engine._EngineClassStub.shutdown(_EngineClassStub.java:173)
    at oracle.reports.server.EngineManager.shutdownEngine(EngineManager.java:1354)
    at oracle.reports.server.JobManager.runJobInEngine(JobManager.java:974)
    at oracle.reports.server.JobManager.runJobLocal(JobManager.java:1779)
    at oracle.reports.server.JobManager.dispatch(JobManager.java:1045)
    at oracle.reports.server.ConnectionImpl.runJob(ConnectionImpl.java:1274)
    at oracle.reports.server._ConnectionImplBase._invoke
```

```
      (_ConnectionImplBase.java:401)
    at com.sun.corba.se.internal.corba.ServerDelegate.dispatch
      (ServerDelegate.java:353)
    at com.sun.corba.se.internal.iiop.ORB.process
      (ORB.java:280)
    at com.sun.corba.se.internal.iiop.RequestProcessor.process
      (RequestProcessor.java:81)
    at com.sun.corba.se.internal.orbutil.ThreadPool$PooledThread.run
      (ThreadPool.java:106)
): Internal error org.omg.CORBA.COMM_FAILURE:   vmcid: SUN  minor code: 201
  completed: No
[2005/6/1 3:38:36:147] State 56004 (EngineInfo:setState): Engine rwEng-0 state is:
  Shutdown
[2005/6/1 3:38:36:147] Info 56047 (EngineManager:remove): Reports Server shut down
  engine rwEng-0
[2005/6/1 3:38:36:147] State 56016 (JobManager:updateJobStatus): Job 17 status is:
  Terminated with error:
REP-56048: Engine rwEng-0 crashed, job Id: 17
[2005/6/1 3:38:36:157] Debug 50103 (JobManager:notifyWaitingJobs): Master job 17
  notify its duplicated jobs.
[2005/6/1 3:38:36:157] Debug 50103 (JobManager:updateJobStatus): Finished updating
  job: 17
[2005/6/1 3:38:36:157] Exception 56048 (): Engine rwEng-0 crashed, job Id: 17
oracle.reports.RWException: IDL:oracle/reports/RWException:1.0
   at oracle.reports.server.JobManager.runJobInEngine(JobManager.java:1009)
   at oracle.reports.server.JobManager.runJobLocal(JobManager.java:1779)
   at oracle.reports.server.JobManager.dispatch(JobManager.java:1045)
   at oracle.reports.server.ConnectionImpl.runJob(ConnectionImpl.java:1274)
   at oracle.reports.server._ConnectionImplBase._invoke
      (_ConnectionImplBase.java:401)
   at com.sun.corba.se.internal.corba.ServerDelegate.dispatch
      (ServerDelegate.java:353)
   at com.sun.corba.se.internal.iiop.ORB.process
      (ORB.java:280)
   at com.sun.corba.se.internal.iiop.RequestProcessor.process
      (RequestProcessor.java:81)
   at com.sun.corba.se.internal.orbutil.ThreadPool$PooledThread.run
      (ThreadPool.java:106)
```

In the engine trace file, the last few lines of the crash trace resemble the following:

```
[2005/6/1 3:38:34:575] (rwfdt:rwfdtprint) Distributing the report
[2005/6/1 3:38:34:585] (rwfdt:rwfdtpredo) running
[2005/6/1 3:38:34:585] (rwfdt:rwfdtpredo) no preformat of pages requested, quit
[2005/6/1 3:38:34:585] (rwfdt:rwfdtni_NextInstance) running
[2005/6/1 3:38:34:595] (rwfdt:rwfdtni_NextInstance) quit
[2005/6/1 3:38:34:595] (rwfdt:rwfdtgcf_GenCachefile) running
[2005/6/1 3:38:34:615] (rwfdt:rwfdtgcf_GenCachefile) Cache file is
  D:\orawin\reports\cache\03564661.htm
[2005/6/1 3:38:34:615] (rwfdt:rwfdtgcf_GenCachefile) quit
[2005/6/1 3:38:34:615] (rwfdt:rwfdtprint) caching output from backend drivers
[2005/6/1 3:38:34:755] (C Engine)
```

> **Note:** The engine trace file ends abruptly whenever the engine crashes.

Action: Identify the report that is causing the engine crash. You can do this by identifying the job ID. In the preceding examples, the engine crashed while running `jobid 17`. In the server trace file, search for the `jobid = 17 Get command line` string.

This line contains the complete command line that includes the report name also. Enable tracing and engine diagnosis. Run the problematic report multiple times to reproduce the crash. When the crash is reproduced, pass on the trace files and diagnosis output to Oracle Support Services for analysis.

D.1.2 Reports Server Activity Generates Error REP-50125

REP-50125 is a common error message issued in multiple situations involving Reports Server:

```
REP-50125: Caught exception: {0}
```

Cause: Oracle Reports has caught an internal exception.

Action: Contact Oracle Support Services for additional assistance.

Problem 1

The Cause and Action in the help topic for REP-50125 do not contain enough information to effectively identify and diagnose the problem.

Solution 1

With Oracle Reports, the following error messages address specific scenarios to provide focused troubleshooting assistance in the Cause and Action exposed in the help topics:

- `REP-56126: Failed to parse server config file {0}`

  Cause: Failed to parse server config file. XML syntax is wrong.

  Action: Correct the server config file and start the server.

- `REP-56127: Failed to decrypt <{0}> element`

  Cause: Decrypt call failed on the element.

  Action: Please make sure encrypted attribute is set properly for the element.

- `REP-56128: Failed to initialize {0} destination. Nested Exception: {1}`

  Cause: Destination initialization failed.

  Action: Please check and correct the configuration for the destination.

Problem 2

REP-50125 displays when starting up Reports Server.

Solution 2

Refer to Note 289748.1 on My Oracle Support at http://support.oracle.com: Troubleshooting Problems When Starting Up Reports Server.

Problem 3

REP-50125 displays when running report requests.

Solution 3

Refer to Note 290827.1 on My Oracle Support at http://support.oracle.com: Troubleshooting Failed Reports Requests Issued Against Reports Server.

Problem 4

`REP-50125` displays with segmentation violation when starting Reports Server on SLES-8/UnitedLinux 1.0.

Solution 4

With Oracle Reports 10*g* Release 2 (10.1.2), SLES8 and SLES9 were supported. However, Oracle Reports 12c (12.2.1.3.0) does not support UnitedLinux 1.0, so you cannot use this platform to run report requests.

Problem 5

`REP-50125` displays when running reports on Linux with openmotif.

Solution 5

Only openmotif 2.1.30 (not higher) is supported for Oracle Reports 6*i*, 9*i*, and 10*g* on Linux.

D.1.3 Long Running Report Failure with Oracle Reports Servlet

Long running report requests submitted through Oracle Reports Servlet (`rwservlet`) may not succeed or cause crashing/hanging engines and timeouts on dependent AS components.

Problem

A report that runs for a long time with `rwservlet` does not finish.

Solution

Perform the following checks:

- If you are running the report synchronously through your Web browser, verify that the failure is not caused by a timeout on the HTTP server. Submit the same job asynchronously; if it finishes successfully, modify the HTTP server timeout in the application server configuration or consider executing your long running reports asynchronously (which is the suggested method). You can leverage the Reports Server notification feature to inform your users when their job has finished.
- Verify that your overall response time of the server has not had any significant changes, by looking at the Reports Server statistics in reports servlet.
- Verify that our database server is responding in a normal manner. Sometimes database load can have a significant impact on the performance, especially on long running reports.

D.1.4 Fonts Do Not Display Consistently On Different Platforms

Deploying reports on multiple platforms may result in font issues.

Problem

When you deploy a report on multiple platforms, font rendering and mapping is not consistent across all platforms.

Solution

This issue is addressed in the following chapters:

- Chapter 9, "Managing Fonts in Oracle Reports" (in particular, see Section 9.1, "Using Fonts" for an understanding of the font handling mechanism in Oracle Reports).
- Chapter 11, "Using PDF in Oracle Reports" (in particular, see Section 11.1.2, "Font-Related Features").
- Chapter 12, "Font Model and Cross-Platform Deployment" (in particular, see Section 12.2.1, "Font Availability On Different Platforms" and Section 12.2.2, "Fixing Font-Related Issues").

D.1.5 Running Reports on UNIX Platforms Generates REP-56048

`REP-56048` is a common error message issued when running a report (for example, using `rwservlet`, `rwclient`, or through Oracle Forms) on UNIX. The Reports Server passes the job to a Report Engine that is responsible for running the report. The Report Engine crashes, resulting in this error:

```
REP-56048: Engine {0} crashed
```

Cause: Reports Server detected the specified engine crashed.

Action: Reports Server should restart another engine. Report the problem to Oracle Support Services with the test case that causes engine crash.

Refer to the solutions below and to Solution 4 in Section D.1.1, "Hanging Report Requests" to attempt to resolve the problem. If REP-56048 persists, perform the following steps to report the problem to Oracle Support Services:

1. Enable server tracing and logging (see Section 24.2.5, "Tracing Report Execution"). If it is not possible to enable tracing, enable logging alone by setting the `log` element's `option` attribute to `failedJobs` in the *`rwserver`*`.conf` file (see Section 7.2.1.12, "log"). When you enable logging, you can see the failed job reports in the `reports.log` file. Identify the report that is failing or causing the engine to crash.
2. Enable engine diagnostic logging by modifying the `engine` element to include the `diagnosis` property in the *`rwserver`*`.conf` file (see "Properties" in Section 7.2.1.8, "engine"), then run the report that you identified in Step 1 to reproduce the crash.
3. Report the crash to Oracle Support Services with the following information:
   - *`rwserver`*`.conf` file.
   - `reports.log` file.
   - Engine diagnostic output when the crash is reproduced.
   - Report definition file so that Oracle Support Services can reproduce the problem.

Problem 1

`REP-56048` displays when running reports on UNIX platforms, and character sets defined in `NLS_LANG` are other than `WE8ISO8859P1` or `IW8ISO8859P8`.

Solution 1

Modify entries in `Tk2Motif.rgb` for mapping Oracle character set names and XLFD's `CHARSET_REGISTRY` to `CHARSET_ENCODING` (the last two fields; for example, `iso8859-1`). For more information, see:

- Section 9.3, "Font Configuration Files"
- Section 18.4.6, "Running with the WE8MSWIN1252 Character Set on UNIX"
- Section 23.7, "Troubleshooting Globalization Issues"
  - Setting Globalization Support Environment Variables
  - Running Oracle Reports in a Japanese Environment on HP-UX

You can also try to run this report with Reports Runtime (`rwrun`) to verify the environment settings before running it through the Report Engine.

Problem 2

`REP-56048` displays when running reports on UNIX platforms, and `DISPLAY` environment variable is not set.

Solution 2

In Oracle Reports, the `REPORTS_DEFAULT_DISPLAY` environment variable removes the dependency on the `DISPLAY` environment variable. By default, `REPORTS_DEFAULT_DISPLAY=YES`. Make sure that `REPORTS_DEFAULT_DISPLAY` has not been set to `NO`.

See Section B.1.43, "REPORTS_DEFAULT_DISPLAY".

Problem 3

`REP-56048` displays when tracing is enabled while running a big report.

Solution 3

If tracing is enabled, Reports Engine might crash for reports with large output. This may be due to the size of the trace file and that there was insufficient disk space, memory, or processor capacity available to create it. To avoid this error, enable engine diagnostic logging only by modifying the `engine` element to include the `diagnosis` property in the *`rwserver`*`.conf` file (see "Properties" in Section 7.2.1.8, "engine"), when diagnostic information is required to troubleshoot a problem with a report. You can also restrict the trace file generated using the `traceModule` attribute of the `trace` element in the *`rwserver`*`.conf` configuration file.

For general information about tracing, see Section 24.2.5, "Tracing Report Execution".

Problem 4

`REP-56048` displays when `DISTRIBUTE=YES` on UNIX.

Solution 4

Distribution fails on UNIX if the `PRINTER` environment variable is not set to a valid printer when one of the destinations specified in the distribution file is a printer. Set the following environment variables:

```
PRINTER=printer_name; export PRINTER
TK_PRINTER=printer_name; export TK_PRINTER
TK_PRINT_STATUS="echo %n is valid"; export TK_PRINT_STATUS
```

```
TK_PRINT=echo; export TK_PRINT
```

In `ORACLE_INSTANCE/config/FRComponent/frcommon/guicommon/tk/admin/uiprint.txt`, add the following line:

```
printer_name:PostScript:1:test:default.ppd:
```

For more information on report distribution, see Chapter 20, "Creating Advanced Distributions".

Problem 5

`REP-56048` displays when printing a report on UNIX.

Solution 5

Oracle Reports uses the shell script `rwlpr.sh` for printing on UNIX. Directly modifying this file is not supported. Please contact Oracle Support Services for assistance.

For more information on printing on UNIX, see Chapter 10, "Printing on UNIX with Oracle Reports".

Problem 6

`REP-56048` displays when running a report containing graphics on UNIX.

Solution 6

This error may result if Oracle Reports is linked against a version of Motif other than the operating system's default. Refer to the Oracle Reports chapter in the *Oracle Fusion Middleware Release Notes for Microsoft Windows* for the correct version of Motif to which to link.

Problem 7

`REP-56048` displays when generating delimited report output for a matrix report.

Solution 7

Generate the report output to DelimitedData (`DESFORMAT=DELIMITEDDATA`) or spreadsheet (`DESFORMAT=SPREADSHEET`) output instead of Delimited. DelimitedData supports large reports, but the output in Microsoft Excel displays only data (as defined by the report data model), no layout information. To generate report output that preserves the formatting defined in report layout, use the output format `DESFORMAT=SPREADSHEET`.

For more information on delimited and spreadsheet output, see "About delimited output" and "About spreadsheet output" in the *Oracle Reports online Help* (and also in the "Advanced Concepts" chapter in the *Oracle Reports User's Guide to Building Reports* manual). Also see Section A.5.27, "DESFORMAT".

Problem 8

`REP-56048` displays when running a report through the Reports Engine when none of the previous solutions resolve the problem.

Solution 8

This error may be related to your environment settings or caused by the report itself. By checking the Oracle Reports Servlet (`rwservlet`) `showjobs` page for your Reports Server, you should be able to determine the job that resulted in the error. If you are on a UNIX machine, there should be a core dump created in your environment. To facilitate searching for related bugs, extract a stack trace from this core dump. Note that the executable should be `java` rather than `rwrun`.

In previous releases, to get a stack trace from the core file, you ran a debugger on the runtime component. This is still applicable if you are able to reproduce the problem with only the `rwrun` component. However, if the crash occurs only through the Reports Server, then the engine will be called using a Java wrapper, and you must run the debugger on the Java executable. This will automatically load any Oracle Reports libraries.

For example:

```
dbx java core
```

This example is for `dbx`. Once you have the stack trace, you will be able to search My Oracle Support at `http://support.oracle.com` for any related issues using the last few calls in the stack.

If a particular job seems to be causing the problem, then the next step would be to try running that report with `rwclient` and `rwrun`. Running with `rwclient` removes the Web component from the environment. Running with `rwrun` is equivalent to bypassing the Reports Server and running with just the engine.

D.1.6 Font Issues with Right-to-Left Languages

Bidirectional support enables you to display report output in either a left-to-right or right-to-left orientation, depending on the requirements of your audience. Font issues with right-to-left languages generate imperfect report output.

Problem

Misalignment of right-aligned text and limitations requiring fixed width fonts.

Solution

Improvements to PDF output with font subsetting enabled for languages that read right to left (such as Hebrew and Arabic), ensure that text will be accurately right-aligned. However, on UNIX platforms, you may see some misalignment for right-aligned text.

To resolve font issues related to right-to-left text, refer to the information in the following sections:

- Section 23.4, "Bidirectional Support" discusses the options available to you in designing reports for right-to-left languages.
- Section 11.3, "Generating a Bidirectional (BiDi) PDF File" outlines the steps involved in generating a PDF file for bidirectional (BiDi) languages.
- Section B.1.29, "REPORTS_ALLOW_DB_CONNECT_STRING" This environment variable allows you to use DB connection strings in the userid parameter.
- Section B.1.30, "REPORTS_BIDI_ALGORITHM" describes the use of this environment variable, which switches the bidirectional (BiDi) layout algorithm for BiDi languages (for example, Arabic or Hebrew).

- Section 12.5, "Generating Multibyte PDF Output" includes an example of a workaround for fixed width font on UNIX.

D.1.7 Errors When Running Reports from Oracle Forms Using RUN_REPORT_OBJECT

The most secure approach for calling Oracle Reports from Oracle Forms on the Web is to use Oracle Reports Services in combination with RUN_REPORT_OBJECT. For detailed information about using RUN_REPORT_OBJECT to call Oracle Reports from Oracle Forms, refer to the *Oracle Application Server 10g Integrating Oracle Reports in Oracle Forms Services Applications* white paper on OTN (http://otn.oracle.com/products/forms/pdf/10g/frm10gsrw10g.pdf).

Also refer to *Forms Services Deployment Guide*.

Problem 1

RUN_REPORT_OBJECT generates the following error:

```
FRM-41214: Unable to run report.
```

Solution 1

When deploying a report over the Web with output to be shown in the browser window, DESTYPE should be set to CACHE, not SCREEN or PREVIEW. To display the output of report in the browser use WEB.SHOW_DOCUMENT, rather than RUN_REPORT_OBJECT.

Problem 2

RUN_REPORT_OBJECT generates the following error:

```
FRM-41213: Unable to connect to the report server server_name.
```

Solution 2

Check the following:

- Ensure that the Reports Server being referenced in the RUN_REPORT_OBJECT code is started. RUN_REPORT_OBJECT will not automatically start the in-process Reports Server if it is not yet started.
- Make sure that the Oracle WebLogic Server instance for Oracle Reports is started.
- Make sure that parameters being passed to RUN_REPORT_OBJECT have no spaces in their values, or are enclosed in single quotes.

Problem 3

RUN_REPORT_OBJECT generates the following error:

```
REP-503 You did not specify the name of a report.
```

Solution 3

Make sure that the report name is specified in the Property Inspector of the Report object in the Oracle Forms Object Navigator.

Problem 4

Unable to run report from a form and pass parameter from a form to the report.

Solution 4

Check the following:

- Make sure that you are passing the parameters in the proper format from Oracle Forms. the initial value for the parameter is specified in the Property Inspector of the parameter in Reports Builder.
- Make sure that the initial value for the parameter is specified in the Property Inspector of the parameter in Reports Builder.
- Try passing the command line in a Before Report trigger or by using the report Parameter Form.

If the issue persists, enable report tracing (see Section 24.2.5, "Tracing Report Execution") to pinpoint the source of the problem.

Problem 5

Using a report Parameter Form (`PARAMFORM=YES`) in conjunction with `RUN_REPORT_OBJECT` fails with "`Internal Server Error`".

Solution 5

Refer to the *Oracle Forms Services - Using Run_Report_Object() to call Reports with a parameter form* white paper on OTN (`http://otn.oracle.com/products/forms/pdf/10g/frmrepparamform.pdf`).

D.1.8 Displaying Report Output in Microsoft Excel

Generating a report to delimited output to display in Microsoft Excel is a common requirement, which can be accomplished in a number of ways. Users are often unsure of which method to choose.

Problem

Which delimited output solution is best for given requirements?

Solution

Depending on your report definition and output display requirements, choose the appropriate method for generating your report to delimited output for Microsoft Excel:

- *Requirement*: You have a paper layout report, which you want output to Microsoft Excel, but do not need rich formatting of the report layout.

  Output Solution: Generate your report to delimited output:

  - `DESFORMAT=DELIMITED`
  - `DESFORMAT=DELIMITEDDATA` (for use when you have problems running large volume reports with `DELIMITED`)

- *Requirement*: You have a paper report, which you want to output to Microsoft Excel, including the rich formatting of the report layout.

  Output Solution: Generate your report to spreadsheet output:

  - `DESFORMAT=SPREADSHEET`

- *Requirement*: You have a paper report, which you want to output to Microsoft Excel, including the rich formatting of the report layout. Additionally, you would like to deploy your report as a JSP.

  Output Solution: Since this is a JSP report, you cannot directly generate to a `.xls` file (`DESTYPE=FILE`),but you can save the output that displays in your browser as a .xls file. Refer to the *Oracle Reports User's Guide to Building Reports* manual to implement this solution using `<rw:include>`: Chapter 29, "Deploying a Web Layout Report to Microsoft Excel Output".

- *Output Requirement*: You have a JSP-based Web report, which you want to output to Microsoft Excel.

  Solution: Since this is a JSP report, you cannot directly generate to a .xls file (`DESTYPE=FILE`), but you can save the output that displays in your browser as a .xls file. Refer to the chapter "Building a report for Spreadsheet Ouput" in the *Oracle Reports User's Guide to Building Reports* manual, available on the Oracle Technology Network.

For detailed information on spreadsheet output and delimited output, see the *Oracle Reports online Help* and *Oracle Reports User's Guide to Building Reports* manual.

D.1.9 Report Containing User Exit Fails on UNIX

User exits may exist in reports developed in prior releases of Oracle Reports.

> **Note:** With Oracle Reports 10*g*, you can call Java methods using the `ORA_JAVA` built-in package and the Java Importer. This reduces the need to have user exits in a report and allows for a more open and portable deployment. You may also use the `ORA_FFI` built-in package, which provides a foreign function interface for invoking C functions in a dynamic library. With the availability of these built-in packages, the use of user exits is deprecated in Oracle Reports, though makefiles are still be supplied to permit you to continue to work with existing user exits.

Problem

A report that contains a user exist fails when run on UNIX.

Solution

On UNIX, Reports Builder (`rwbuilder`) and Reports Runtime (`rwrun`) dynamically load the user exit library to successfully run reports that contain user exits. When running reports through Reports Server (`rwserver`), you must add the following environment variable in `rwengine.sh` to load the user exit library:

```
LD_PRELOAD=librw.so:user_exit_library; export LD_PRELOAD
```

D.1.10 Printing and Font Errors When Using In-process Reports Server

The in-process Reports 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 Reports Server is logged on as the Local System.

Problem

Any of the following:

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

  ```
  http://myrepsrvr.example.com:7777/reports/rwservlet?report=myrep.rdf&destype=printer&desformat=html
  ```

  results in the following error:

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

- Deploying reports containing Oracle6*i* Graphics (OGD) graphics causes Reports Server to stop responding.
- Font alignment problems in a PDF file output from an in-process Reports Server.

Solution

To work around all these issues:

1. Open the Windows registry using a registry editor (for example, `regedit.exe`). Create a backup of the registry before you edit 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 every time you alter the value of the Default Printer.
>
> This workaround will not work on an OPMN-managed Reports Server.

The pros and cons of running an in-process Reports Server are explored in Chapter 2, "Understanding the Oracle Reports Services Architecture". For additional information, see Section 7.3, "Oracle Reports Servlet Configuration File" (server and inprocess parameter descriptions).

D.1.11 Runtime execution of Reports shifts down a record in the placeholder coulmn values

Placeholder column values shift down a record when the report is executed in the Reports Runtime. However, if the report is processed in the Reports Builder, then the report output is formatted correctly. Reports Runtime requires a dependency to be established between the placeholder coulmn and the formula column.

Problem

Reports Runtime execution causes the value to shift down a record in the following cases:

- When a report contains a placeholder column(s) with the value(s) set from a formula column.
- When a report contains a placeholder column where the value(s) is derived from a formula column procedure call.

Solution

To resolve this issue, perform the following steps:

1. From the Oracle Report data model, select the property palette for the placeholder column. For example, the column that you selected is CP_1.
2. Double-click the PL/SQL Formula column property.
3. Enter an `srw.reference` from the placeholder column to the formula column, as shown in the following example:

```
function CP_1Formula return Number is
Begin
srw.reference (:CF_1);
end;
```

where, CF_1 is the Formula column, and `srw.reference` establishes the dependency between the placeholder column and formula column.

4. Select **Compile**.
5. Repeat the above steps for other placeholder columns, as required.
6. Save the report and rerun it in Reports Runtime. The placeholder column will no longer shift down a record.

D.2 Diagnosing Performance Problems

For information about how you can improve your report execution time and streamline the overall performance, see Chapter 24, "Diagnosing and Tuning Oracle Reports".

For information about using Oracle Reports tracing options to trace and diagnose problems, including performance-related problems with Oracle Reports, refer to Section 24.2.5, "Tracing Report Execution".

D.3 Diagnosing Font Problems

For common problems and solutions when using fonts in UNIX machines, see Chapter 12, "Font Model and Cross-Platform Deployment".

D.4 Diagnosing Printing Problems

For common problems and solutions when printing your reports in UNIX, see Section 10.9, "Frequently Asked Questions" in Chapter 10, "Printing on UNIX with Oracle Reports".

D.5 Diagnosing JDBC PDS Problems

For common problems and solutions when using the JDBC PDS, see Section 14.1.4, "TroubleShooting Information" in Chapter 14, "Configuring and Using the Pluggable Data sources".

D.6 Diagnosing Oracle Portal Problems

For common problems and solutions when publishing your reports to Oracle Portal as a portlet or item link, see Section 16.5, "Troubleshooting Information" in Chapter 16, "Deploying Reports in Oracle Portal":

D.7 Diagnosing Globalization Problems

For common globalization problems and solutions in Oracle Reports, see Section 23.7, "Troubleshooting Globalization Issues" in Chapter 23, "Implementing Globalization and Bidirectional Support".

D.8 Diagnosing Oracle Reports Bridge Problems

Issues in communication across subnets while using the built-in broadcast mechanism may be related to the Oracle Reports bridge.

problem 1

The Oracle Reports bridge trace file contains the following information:

```
[2005/12/8 1:45:7:339] Info 50103 (BridgeConnection:getResponsePacket):
Getting response object from remote bridge
usunrao06.example.com/130.35.37.76:14011
[2005/12/8 1:45:8:340] Debug 50103 (BridgePacketHandler:handleRequestPacket): Got
response from remote subnet null
[2005/12/8 1:45:8:991] Error 64013 (BridgeConnection:getResponsePacket):
Bridge failed to serve the requestjava.net.ConnectException: Connection refused:
connect
```

Solution 1

The remote Oracle Reports bridge is not running. Start the Oracle Reports bridge, as described in Section 5.2, "Starting, Stopping Reports Bridges from the Node Manager using script"

Problem 2

```
[2005/12/8 1:50:34:219] Info 50103 (BridgeConnection:getResponsePacket):
Getting response object from remote bridge
usunrao06.example.com/130.35.37.76:14011
[2005/12/8 1:50:34:469] Debug 50103 (NetworkUtility:write): Writing ......
[2005/12/8 1:50:34:469] Debug 50103 (NetworkUtility:read): Reading ......
[2005/12/8 1:50:35:220] Debug 50103 (BridgePacketHandler:handleRequestPacket): Got
response from remote subnet null
```

Additionally, the Oracle Reports bridge trace file in the remote subnet (the subnet where Reports Server is located) contains the following information:

```
[2005/12/8 2:36:59:997] Debug 50103 (Multicast:registerReceiver): Packet handler
registered
[2005/12/8 2:37:1:16] Info 65003 (NetworkUtility:getIOR): Request timed out
[2005/12/8 2:37:2:19] Info 65003 (NetworkUtility:getIOR): Request timed out
[2005/12/8 2:37:3:30] Info 65003 (NetworkUtility:getIOR): Request timed out
[2005/12/8 2:37:3:31] Debug 50103 (NetworkUtility:getIOR): No response from server
retuning null ior
[2005/12/8 2:37:3:32] Info 50103 (Multicast:registerReceiver): Packet handler
unregistered
[2005/12/8 2:37:3:32] Debug 50103 (NetworkUtility:write): Writing ......
```

Solution 2

Reports Server is not running on the remote subnet. Start Reports Server, as described in Section 5.1, "Starting and Stopping Reports Server"

For comparison, the following sample output shows the Oracle Reports bridge trace for the scenario where the Oracle Reports bridge successfully discovers Reports Server:

```
[2005/12/8 4:4:6:700] Info 50103 (BridgeConnection:getResponsePacket):
Getting response object from remote bridge
usunrao06.example.com/130.35.37.76:14011
[2005/12/8 4:4:6:950] Debug 50103 (NetworkUtility:write): Writing ......
[2005/12/8 4:4:6:950] Debug 50103 (NetworkUtility:read): Reading ......
[2005/12/8 4:4:7:932] Error 50103 (BridgeConnection:run): Got response
[2005/12/8 4:4:7:942] Debug 50103 (BridgePacketHandler:handleRequestPacket):
Got response from remote subnet Response Packet -
ServerName = vin
CorrelationID = 1134056309492
SenderID = ServerName: vinVMID: 7e444dbc56c79b06:f9c40:10809ddbdaf:-8000
Duplicate = false
Type = FULL
Add. Info = Type = server : Host = usunrao06.example.com
```

Problem 3

Reports Server is running on the remote subnet, Oracle Reports bridges are running on both the subnets and are configured properly, but the Oracle Reports clients are not able to connect to the remote Reports Server.

This may be caused by an improper setting of timeout value in the bridge configuration file. In this case, the Oracle Reports bridge in the local subnet (the subnet where the client is located) may time out before the Oracle Reports bridge in the remote subnet can respond.

Solution 3

Increase the timeout value in the Oracle Reports bridge configuration file (`rwbridge.conf`). For example:

```
<bridge version="10.1.2" port="14011" timeout="2000">
```

The Oracle Reports bridge in the local subnet waits for the timeout period to get a response from the Oracle Reports bridge in the remote subnet. The Oracle Reports bridge may time out if the network connectivity is slow.

Additionally, increase the timeout value in the network configuration file (`rwnetwork.conf`) to avoid the timeout of Oracle Reports clients before the Oracle Reports bridge responds. For example:

```
<multicast channel="228.5.6.7" port="14021" timeout="1000" retry="3"/>
```

In general, perform the following steps to set the Oracle Reports bridge timeout. It is assumed that both bridges are configured properly.

1. Set a very high timeout value for the Oracle Reports bridge in the local subnet.
2. Start both Oracle Reports bridges.
3. Start Reports Server (in the remote subnet).
4. In the local subnet, run the `rwdiag` utility with the following command:

   ```
   rwdiag -find server_name
   ```

   This command prints the time that the Oracle Reports bridge takes to discover the remote Reports Server. For example:

```
D:\orawin\reports\conf>rwdiag -find vin
Broadcast mechanism used to locate servers
----------------------------------------
Channel address = 228.5.6.7
Channel port = 14021
'vin' found in the network
Time taken - 1181 milliseconds
Name = vin : Type = server : Host = usunrao06.example.com
```

In this example, the Oracle Reports bridge has taken 1181 milliseconds to discover the remote Reports Server.

5. Estimate the timeout value for the Oracle Reports bridge located in the local subnet as follows:

   `timeout = 1181 * 1.3`
   which is about 1500 milliseconds.

6. Set the timeout value in the `repbrg_bridgename.conf` file. For example:

   ```
   <bridge version="10.1.2" port="14011" timeout="1500">
   ```

7. Confirm that the timeout value in the `rwnetwork.conf` file complies with the following:

   `timeout (in rwnetwork.conf) * retry (in rwnetwork.conf) > timeout (in repbrg_bridgename.conf)`
   For example:

   `timeout (in rwnetwork.conf) * 3 > 1500`
   Therefore, the value of timeout in `rwnetwork.conf` should be `1500` or higher

Problem 4

You want to find out which Oracle Reports bridges are configured and running in the subnet.

Solution 4

Use the `rwdiag` utility to locate all Reports Servers and Oracle Reports bridges. For example, run the following command:

```
rwdiag -findall
```

This command generates output similar to the following example:

```
C:\>rwdiag -findall
Broadcast mechanism used to locate servers
----------------------------------------
Channel address = 228.5.6.7
Channel port = 14021
(1) Name = bugupdate : Type = server : Host = strep15.idc.oracle.com
(2) Name = rep_supadhya-pc_frhome1 : Type = server : Host =
supadhya-pc.idc.oracle.com
(3) Name = vinod : Type = server : Host = strep10.idc.oracle.com
(4) Name = rep_strep10 : Type = server : Host = strep10.idc.oracle.com
(5) Name = abc : Type = bridge : Host = strep12.idc.oracle.com
(6) Name = rep_stfrm08_frhome1 : Type = server : Host = stfrm08.idc.oracle.com
(7) Name = rep_stport79_as101202mid : Type = server : Host =
stport79.idc.oracle.com
(8) Name = rep_iwinreb20_0508041930_bif : Type = server : Host =
iwinreb20.example.com
C:\>
```

For more information on the rwdiag utility, refer to Appendix E, "Reports Server and Bridge Diagnostic Utility".

D.9 Need More Help?

You can find more solutions on My Oracle Support at `http://support.oracle.com`. If you do not find a solution for your problem, your Oracle representative can log a service request.

To help Oracle Support Services troubleshoot the problem, perform the following steps:

1. **Trace report execution**, as described in Section 24.2.5, "Tracing Report Execution".
2. **Contact Oracle Support Services**. To help Oracle Support Services troubleshoot the problem, provide a zip file containing the trace information, deployment scenario, or sample report output, if required.

See Also:

- *Oracle Fusion Middleware Release Notes for Microsoft Windows*, available on the Oracle Technology Network (OTN).

E

Reports Server and Bridge Diagnostic Utility

This appendix describes the arguments and usage of the Reports Server and bridge diagnostic utility, `rwdiag`.

- Overview of rwdiag
- Command Line Syntax

For troubleshooting scenarios and diagnosis, see Section D.8, "Diagnosing Oracle Reports Bridge Problems".

E.1 Overview of rwdiag

`rwdiag` is a utility used to find Reports Servers and bridges on the network, and monitor packets broadcast on the network by the Reports Server and its clients. It is also helpful for choosing optimal settings for `${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/rwnetwork.conf` and for bridge timeout values. rwdiag is similar to the `osfind` utility provided by the Borland VisiBroker ORB, which has been replaced by the JDK ORB in Oracle Reports.

> **Note:** Oracle Reports replaces the use of Borland's Visibroker with Sun Microsystems' industry-standard Java Developer's Kit Object Request Broker (JDK ORB). The JDK ORB provides support for Reports Server requests from clients across subnets, and enables the broadcast mechanism for dynamic Reports Server discovery both within a subnet and across subnets.

You can invoke rwdiag with one of two scripts depending upon your operating platform:

For Microsoft Windows:

```
$DOMAIN_HOME\reports\bin\rwdiag.bat
```

For UNIX:

```
$DOMAIN_HOME/reports/bin/rwdiag.sh
```

E.1.1 Examples

The sections that follow provide a series of examples illustrating the use of rwdiag.

E.1.1.1 Example 1

This command line tries to find a Reports Server or bridge named `abc` on the network with the default search timeout of 10 seconds.

```
rwdiag.bat -find abc
```

This command returns a success message, name, type, host name, and the time taken, if `abc` is found on the network. If a naming service is used as the discovery mechanism for Oracle Reports, only the success message would be returned as the host name would be unavailable to the utility.

E.1.1.2 Example 2

This command tries to find a Reports Server or bridge named abc on the network with a search timeout of 5 seconds.

```
rwdiag.bat -find abc -timeout 5
```

E.1.1.3 Example 3

This command tries to find a Reports Server or bridge named abc on the network using the settings in the configuration file xyz.conf.

```
rwdiag.bat -find abc -conf xyz.conf
```

> **Note:** If the network configuration file is stored in a location other that the default location, you must specify the complete path of the file.

Following are the contents of xyz.conf:

```
<?xml version = '1.0' encoding = 'ISO-8859-1'?>
<!DOCTYPE discoveryService SYSTEM "file:c:\orawin\reports\dtd\rwnetworkconf.dtd">
<discoveryService>
<multicast channel="105.2.3.8" port="35078" timeout="1000" retry="3"/>
<!--namingService name="Cos" host="localhost" port="9999"/-->
</discoveryService>
```

Notice how the channel address and port number are picked up from the configuration file. If for some reason abc were running on another port, it would not be found.

E.1.1.4 Example 4

This command tries to find all Reports Servers and bridges on the network.

```
rwdiag.bat -findAll
```

With a broadcast mechanism, all information is provided. If a naming service is used as the discovery mechanism for Oracle Reports, host information is unavailable.

E.1.1.5 Example 5

This command monitors all packets broadcast on the network by the Reports Servers and their clients, and prints the packet information on the screen. The monitoring stops when you press q and Enter.

```
rwdiag.bat -monitor
```

E.1.1.6 Example 6

This command monitors all packets broadcast on the network by the Reports Servers and their clients, and saves the packet information to the log file, c:\log.txt. The monitoring stops when you press q and Enter.

```
rwdiag.bat -monitor -log c:\log.txt
```

E.2 Command Line Syntax

rwdiag includes keywords that enable you to do the following:

- Locate a Reports Server or bridge running on the network.
- List all running Reports Servers or bridges on the network.
- Monitor packets on the network broadcast by Reports Servers or clients.

E.2.1 Syntax

```
rwdiag.bat | rwdiag.sh {-find serverName | -findAll |
-monitor [-log log_file_name]} [-conf config_file_name] [-timeout seconds]
```

Where

`rwdiag.bat` is the script for Microsoft Windows.

`rwdiag.sh` is the script for UNIX.

`-find` *`serverName`* finds the Reports Server with given name running on the current network.

`-findAll` finds and lists all Reports Servers and bridges on the network.

`-monitor` lists the packets broadcast on the network. To stop monitoring, press q and Enter. This option is not supported when the discovery mechanism specified in the network configuration file is a naming service.

`-log` *`log_file_name`* specifies a log file to which the monitor output is written. If not specified, the monitor output is displayed on the screen. The file name can be an absolute path. If just a file name is specified, the log file is created in the current folder.

`-conf` *`config_file_name`* specifies a custom configuration file. If not specified, `rwnetwork.conf` is the default file name. The settings such as discovery mechanism (broadcast or naming service) and port numbers are taken from this file. The utility assumes the configuration file is located in *`${DOMAIN_HOME}/config/fmwconfig/components/ReportsToolsComponent/<reports_tools_name>/`*. If a non-existent file is specified, the file is created with the default settings in `rwnetwork.template`.

`-timeout` *`seconds`* specifies the timeout value in seconds. If not specified, the default value is 10 seconds. Timeout is the length of time the client waits for a response from the server after broadcasting the request packet. This option is ignored when the discovery mechanism specified in the network configuration file is naming service.

E.2.2 Usage Notes

- The host information is not available when using a naming service discovery mechanism.
- Time taken to locate the server is not displayed for a naming service discovery mechanism because the lookup is based upon the Reports Server name in the

naming service. The utility does not need to await response from the server. Hence, the time taken is not relevant for a naming service.

- Bridges cannot be located using a naming service because they do not bind to the naming service. Only Reports Server implementations are bound to the naming service.
- The timeout value in the configuration file is ignored. Only the value specified in the command line is taken into account. If not specified in the command line, the default value is 10 seconds.
- If the Reports Server you try to locate is not found, the utility generates a `REP-50504` message, which states that the server was not found.