

Integrating Oracle9iAS Reports Services in Oracle9iAS Forms Services

An Oracle Whitepaper

Covers

- *Oracle9i Forms and Oracle9i Reports in Oracle9iAS
Release 2 and Oracle9iDS Release 2*
- *Oracle9iAS Forms and Reports Single Sign-On*

December 2002

Integrating Oracle9iAS Reports Services in Oracle9iAS Forms Services

Introduction.....	4
Audience addressed in this paper	4
Checklist to run integrated Reports in Oracle9iAS Forms Services.....	5
Oracle9i Forms	5
Understanding Oracle9i Forms Developer	6
Understanding Oracle9iAS Forms Services	6
The formsweb.cfg configuration file	6
The default.env environment file	7
The Forms Servlet	8
The Forms Listener Servlet.....	8
OC4J and Mod_OC4J	9
Oracle9iAS Forms Services configuration in Oracle9iDS	9
Oracle9iAS Forms Services configuration in Oracle9iAS.....	9
Oracle9i Reports	9
Understanding Oracle9i Reports Developer.....	9
Reports with paper layout	10
Reports with Web layout.....	10
Understanding Oracle9iAS Reports Services.....	10
The Reports Servlet.....	11
The Reports CGI.....	11
The Reports Server.....	11
The rwservlet.properties configuration file.....	12
The <Reports Server>.conf configuration file	12
The cgicmd.dat file	12
Oracle9iAS Reports Services configuration in Oracle9iDS.....	13
Oracle9iAS Reports Services configuration in Oracle9iAS	13
Single Sign-On Integration (available with Oracle9iAS).....	13
mod_osso	14
Single Sign-On in Forms.....	14
Enabling single sign-on in Forms.....	15
Single Sign-On in Reports	16
Enabling single sign-on in Reports	18
Enabling access control for Reports Services	18
Disabling single sign-on in Reports	18
Disabling Oracle Reports access control	18
How does single sign-on relate to Reports integrated in Forms?	19

Forms / Reports integration single sign-on matrix.....	19
Calling Oracle Reports from Forms On the Web	20
Introducing the RUN_REPORT_OBJECT built-in.....	20
Using parameterlists in RUN_REPORT_OBJECT	22
Using the RUN_REPORT_OBJECT built-in	23
RUN_REPORT_OBJECT example	23
Passing Forms parameter lists in RUN_REPRORT_OBJECT.....	26
Calling Reports that display a parameter form.....	26
Using the WEB.SHOW_DOCUMENT built-in	31
Syntax.....	31
Example - WEB.SHOW_DOCUMENT ()	32
Example - WEB.SHOW_DOCUMENT() & relative addressing ..	33
Example - Obfuscating the user credentials in the URL.....	33
Example – Reports SSO and WEB.SHOW_DOCUMENT()	35
Printing Reports output on the Web.....	36
Forms Migration Assistant (FMA).....	36
Known issue with rp2rro.pll and workaround.....	37
Summary	38
Appendix: Usability matrix	39
Appendix B: Passing Forms parameter lists.....	39
Forms Parameter list example	40
Appendix C: Troubleshooting.....	41

Integrating Oracle9iAS Reports in Oracle9iAS Forms Services

Another version of this paper, covering the integration of Oracle Reports6i in Oracle Forms Services 6i, is available at otn.oracle.com/products/forms

INTRODUCTION

Oracle9i Application Server Release 2 (Oracle9iAS) and Oracle9i Developer Suite Release 2 (Oracle9iDS) contain Oracle9i Forms and Oracle9i Reports. Starting with this release of both products, there is no longer a client/server runtime environment available, which means that Forms and Reports applications must be run on the Web.

This paper focuses on the integration of Oracle9i Reports in Oracle9iAS Forms Services and discusses in great detail *how* to call Oracle9i Reports integrated in Oracle9i Forms applications. Throughout this paper Oracle9i Application Server Release 2 is referred to as Oracle9iAS, while Oracle9i Developer Suite Release 2 is referred to as Oracle9iDS. If the release number of Oracle Forms and Oracle Reports is not explicitly mentioned, then implicitly the Oracle9i Forms and Oracle9i Reports version is meant. After reading this whitepaper you will

- Know about the configuration files involved and how to work with them
- Know how to use the Forms RUN_REPORT_OBJECT() built-in, replacing RUN_PRODUCT() for calling integrated Oracle Reports
- Know how to use the WEB.SHOW_DOCUMENT() built-in to download the Reports output from the middle tier
- Understand Single Sign-On for Oracle9iAS Reports Services and Oracle9iAS Forms Services
- Know about existing problems and how to work around them

Audience addressed in this paper

The audiences addressed in this paper are customers and technical consultants who know the Forms Reports integration in client/server environments and that plan to move their applications to the Web using Oracle9i Forms and Oracle9i Reports in Oracle9iAS Release 2 and Oracle9iDS Release 2.

Customers who already run their Forms applications on the Web will learn about the differences between running integrated Reports in Forms6i and running them in Oracle9i Forms. Also covered is the format change of the Oracle9i Reports

'getjobid' parameter ensuring that existing Forms6i Web applications with integrated calls to Oracle Reports continue working in Oracle9iAS and Oracle9iDS.

Checklist to run integrated Reports in Oracle9iAS Forms Services

The following list should give you some idea of what needs to be done to successfully run integrated Oracle Reports from Forms on the Web:

- Oracle9iAS Forms does not run in single sign-on mode by default, while Oracle9iAS Reports does. You will need to configure one of them to its opposite.
- Oracle9iAS Forms Services can not run Oracle9iAS Reports that are access controlled.
- RUN_PRODUCT() is no longer supported in Forms to perform calls to integrated Oracle Reports. Use RUN_REPORT_OBJECT() instead.
- The Oracle9iAS Reports Services 'getjobid' parameter, no longer contains the Reports Server name in its value, nor does it allow an equal sign as a delimiter between the parameter name and the value.
- The Reports destypes 'screen' and 'preview' no longer exist. Previewing a Reports output before printing must be done on the Web using destype=cache and desformat=htmlless or desformat=pdf
- There is no local printer support¹. Oracle9iAS Reports Services prints to network printers only, which is common now on the Web.
- Record groups can no longer be passed to Oracle Reports as data parameters in a parameter list. Use a query in the Reports definition file instead.
- Calling a report with its own parameter form displayed does not work out-of-the-box. A workaround for calling such reports is provided in this paper.

All of this is covered in the following sections contained in this paper.

ORACLE9i FORMS

Oracle9i Forms consists of two components: the Oracle9i Forms Developer building component and the Oracle9iAS Forms Services Web deployment component. Oracle9iAS Forms Services is a part of the Oracle9i Application Server Release 2 offering, while Oracle9i Forms Developer is part of Oracle9iDS.

Coming from the client/server paradigm, most Forms applications use one of the following two built-ins to create a report using the Reports client runtime engine:

¹ You can still download the report output to the client Browser and use the Browser's printing facilities.

- RUN_PRODUCT built-in
- RUN_REPORT_OBJECT built-in

With Oracle9i Forms and Oracle9i Reports, the client runtime engine no longer exists, and so all integrated calls from Forms in the Web to Oracle Reports must use Reports Services². There are two Oracle9i Forms built-ins supported for running Reports on the Web:

- RUN_REPORT_OBJECT built-in
- WEB.SHOW_DOCUMENT built-in

Both of these built-ins, and how to use them when calling integrated Oracle Reports in Forms, are explained later in this whitepaper.

Understanding Oracle9i Forms Developer

Oracle9i Forms Developer is the name of the Forms application building environment that is shipped with the Oracle9i Developer Suite.

Understanding Oracle9iAS Forms Services

Oracle9iAS Forms Services is available in Oracle9iAS for production deployments and in Oracle9iDS for testing. The Forms Services components and configuration files provided in Oracle9iDS are the same as those in Oracle9iAS, so that an application developed in Forms Developer works the same in the production environment as in the test environment.

The following main components and configuration files are used when running Forms on the Web

- The formsweb.cfg configuration file
- The default.env environment file
- The Forms Servlet
- The Forms Listener Servlet
- Oracle Containers for J2EE (OC4J)

The formsweb.cfg configuration file

The formsweb.cfg configuration file, located in the forms90\server directory of every Oracle9iDS or Oracle9iAS installation, is read by the Forms Servlet to build the Forms Applet start HTML page. Information in this configuration file is separated into three categories:

² The exception from this rule is for applications that use the Reports background engine for printing only. The Reports background engine is accessible directly from Forms on the server by configuring the Reports_Path variable in the Forms default.env file. Oracle recommends to use Oracle9iAS Reports Services for integrated reporting in Forms.

For an in-depth view of the Forms Services architecture, refer to the whitepaper *Oracle9iForms Services Overview* at otn.oracle.com/products/forms

- System parameters — System parameters are parameters that cannot be specified within the actual application URL. These parameters determine the Applet HTML template file, the default working directory, and the environment file used, unless these files are otherwise specified later in the same configuration file.
- User parameters — User parameters are default parameter settings, like ‘form’, ‘userid’, Applet ‘width’ and Applet ‘height’, which can be overwritten in the application URL or later in the same configuration file. If a parameter is not mentioned in the request URL for an application, its value is instead taken from the default settings.
- Custom application definition — A named configuration that is a logical group of system and user parameters used for one particular application. The name of the configuration appears in the request URL as the value of the config parameter (config=<named configuration>). Parameters that are not included in a named configuration or specified in the URL are taken from the default settings.

```
[reptest]
form=reptest90
userid=scott/tiger@orcl
look&feel=oracle
width=700
height=500
```

Example 1: Custom application definition in formsweb.cfg

You can call the application ‘reptest’ from the Web by issuing

```
http://<hostname>:<port>/forms90/f90ervlet?config=reptest
```

If an extra parameter needs to be passed in then it can be added to the named configuration or appended to the URL

```
http://<hostname>:<port>/forms90/f90ervlet?config=reptest&
separateFrame=true
```

For Forms/Reports integration, you can also use the “otherparams” parameter in the formsweb.cfg file to pass extra information (such as the name of the Reports Server to use) when starting the Forms Web application.

The default.env environment file

The default environment file is specified with the ‘envFile’ parameter in the systems parameters section of the formsweb.cfg file. The ‘default.env’ file determines the environment setting, like Forms90_Path, in which a Forms runtime engine is started. Overwriting the ‘envFile’ parameter in the named configuration

Unlike in client-server configurations, the Reports_Path environment variable does not need to be specified for the Forms Services environment.

section of an application allows you to start different applications with different environment settings:

```
[reptest]
form=reptest90
...
envFile=myRep.env
```

Example 2: Referencing a custom environment file in formsweb.cfg

For Oracle9i Forms/Oracle9i Reports integration, you do not need to specify the Reports_Path for an application as long as the Forms Services can access the Reports Server, which is the default configuration for applications installed with Oracle9iDS or Oracle9iAS.

The Forms Servlet

The Forms Servlet, which acts as the Forms Services Web Interface, is by default accessible through `http://<hostname>:<port>/forms90/f90servlet`.

When calling the Forms Servlet URL to start a Forms application on the Web, you'll need to pass either a custom configuration section specified in the formsweb.cfg file or the complete list of Forms runtime parameters.

The following URLs are all valid:

```
http://<hostname>:<port>/forms90/f90servlet?config=reptest
```

```
http://<hostname>:<port>/forms90/f90servlet?config=reptest&form=reptest
```

```
http://<hostname>:<port>/forms90/f90servlet?form=reptest&userid=scott/tiger@orcl&lookandfeel=oracle
```

The Forms Servlet uses the formsweb.cfg file to generate the application start HTML file, which, in turn, initializes the download of the Forms Java Applet to the client. After this, the Forms Servlet is released to serve other application requests. The Forms Web runtime process is started in the environment specified by the 'default.env' file if not overwritten in the custom application section.

The Forms Listener Servlet

The Forms Listener Servlet dispatches the communication between the Forms client and the Forms user runtime process on the server. The Forms Listener Servlet is defined in the formsweb.cfg file by the serverURL parameter. The default value is `"/forms90/f90servlet"`.

OC4J and Mod_OC4J

Oracle Containers for J2EE (OC4J), the default servlet container in Oracle9iAS and Oracle9iDS, is the runtime environment for the Forms Servlet and the Forms Listener Servlet component.

Mod_OC4J is an Oracle Apache module in Oracle9iAS that routes HTTP Servlet requests to the OC4J engine when running Forms Services on Oracle9iAS.

Oracle9iAS Forms Services configuration in Oracle9iDS

For more information about Oracle9iDS, refer to the Oracle9iDS Release 2 data sheet on otn.oracle.com

To conserve disk space, Oracle9iDS does not contain the Oracle HTTP Listener and its components, but instead uses the integrated HTTP listener in OC4J to runtime-test Forms and Reports Web applications. All Web communication is handled by the OC4J-integrated HTTP Server. The configuration of the Forms runtime environment is the same as in Oracle9i Application Server, except that neither mod_oc4 nor the single sign-on feature is available in Oracle9iDS.

The Forms Web runtime environment is automatically configured when Oracle9iDS is installed. No additional, manual configurations are required to call Reports from Forms applications.

Oracle9iAS Forms Services configuration in Oracle9iAS

For more information about Oracle9iAS, refer to the Oracle9iAS Release 2 data sheet on otn.oracle.com

The Forms Servlet and the Forms Listener Servlet running in OC4J are accessed through mod_oc4j, an Oracle Apache module, when your application is running in Oracle9iAS. Mod_oc4 routes all calls to /forms90/f90servlet and /forms90/i90servlet to OC4J, while downloads of images and Java archive files used within a Forms application are handled by the Oracle HTTP Server.

All required configurations are automatically performed when Oracle9i Application Server is installed. As with Oracle9iDS, there is no need for additional, manual configurations to call Reports from Forms applications.

ORACLE9i REPORTS

Like Forms, Oracle9i Reports is the brand name for two separate components: the Oracle9i Reports Developer building environment and the Oracle9iAS Reports Services deployment environment. Throughout this paper, the terms Reports Services and Reports Server are used interchangeably for the same component. Oracle9i Reports Developer is part of the Oracle9iDS offering and is available on the same platforms as is Oracle9iDS.

Oracle9iAS Reports Services, the Reports Server component, is included in Oracle9i Application Server Release 2.

Understanding Oracle9i Reports Developer

Refer to the *Oracle Reports New Features* whitepaper at otn.oracle.com/products/reports for more information on Oracle9i Reports.

Oracle9i Reports Developer, the building environment for new and existing Reports application modules, contains numerous new features like the added support for Web layouts. Web layouts allow a Reports application developer to

include Reports content into any existing JSP page or to define the complete Reports definition in a stand-alone JSP file, rather than using the rdf file format.

Existing Reports, built with previous versions of Oracle Reports, use the paper layout, the most common layout type for Reports integrated in Forms applications.

Oracle9i Reports Developer has a built-in preview functionality for Reports paper and Web layouts.

Reports with paper layout

The Reports paper layout is the name for the “old” report layout; that is, the only layout available in Reports releases prior to Oracle9i Reports. The dimension of a Reports in paper layout is defined by the size of the paper used to print the report content. Although previous Reports releases supported the display of both HTML and HTMLCSS output in a Web browser, the dimension used was always the size of a sheet of paper.

Reports with Web layout

Before Oracle9i Reports, it was not possible to transfer the output from Reports into an existing Web page, or to create a Reports Web output with a width wider than the margin of a sheet of paper. The new Web layout allows a Reports designer to freely design the page size of the report, with no constraints enforced. Web layouts are defined in Java Server Pages (JSP) files, which can be created either within Reports Developer, without any manual JSP coding, or within Oracle9i JDeveloper, which contains the Oracle9i Reports tag library.

More information about Web layouts in Oracle Reports is provided in the tutorial *Getting started with Oracle9i Reports*, accessible from the Reports9i link on otn.oracle.com/products/reports

Reports Web layouts cannot be called from Forms with the Run_Report_ Object built-in. If you need to call a Reports Web layout from Forms, use the Forms WEB.SHOW_DOCUMENT() built-in with a URL pointing to the jsp page holding the Reports definition.

Understanding Oracle9iAS Reports Services

Oracle Reports Services, the Web deployment environment for Oracle Reports modules, is part of Oracle9i Application Server Release 2. Oracle9iAS Reports Services is also referred to as a *Reports Server*. Both names are used interchangeably throughout this paper. The following main components and configuration files are used when running Reports on the Web:

- The Reports Servlet (*rwservlet* for paper layouts)
- The Reports CGI (for backward compatibility)
- The Reports Server process
- Rwservlet.properties configuration file
- <Reports Server>.conf configuration file
- The cgicmd.dat file

The Reports Servlet

The default Oracle9iAS Reports Services Web interface is a servlet, *rwervlet*, which dispatches incoming Reports HTTP requests to the Reports Server engine and performs single sign-on authentication where required.

Integrated calls to Oracle Reports in Forms use the servlet either to request a report or to download the resulting Reports output to the Forms client browser.

The Reports CGI

The Oracle9i Reports Common Gateway Interface (CGI) performs the same tasks as the Reports Servlet, but is provided for backward compatibility only. If you have an existing Forms Web application with integrated calls to the Oracle Reports Server that uses the CGI interface, then be aware that the CGI will go away in future releases with further notice.

Oracle recommends using the Reports Servlet. If your business requires single sign-on, the Reports Servlet is mandatory.

The Reports Server

The Reports Server process is a management instance for multiple Reports runtime engines. There are two types of Reports servers available with Oracle9i Reports: the Reports 'in process' server and the Reports server running in an extra process.

The Reports 'in process'³ server is started through the Reports Servlet *rwervlet* the first time a report is requested. The 'in process' server is used whenever a Reports Server name is not provided within the server parameter of the request.

The Reports 'in process' server can be started and accessed only through the Reports Servlet *rwervlet*, and it runs in the same process as the servlet. The name of the 'in process' server is `rep_<hostname>`, and, once started, its configuration filename and location is `reports/conf/rep_<hostname>`.

The second type of Reports Server runs in its own process, similar to the Reports Server process in Reports6i. The Reports Server can be started by the following command, accessible from the `\bin` directory of the Oracle9iDS or Oracle9iAS installation.

To access a Reports Services from the Forms `RUN_REPORT_OBJECT()` built-in, you should not have an underscore in the Reports Server name

```
rwserver -install <server_name> (Windows)
```

or

```
rwserver.sh server=<server_name> batch=yes & (Unix)
```

³ The Reports 'in process' server cannot be used with `RUN_REPORT_OBJECT()` in Forms, but can be used instead with the `WEB.SHOW_DOCUMENT()` built-in.

Any value can be used for <server_name>, as long as the name selected is unique in the accessible network. To work with RUN_REPORT_OBJECT() in Forms, the Reports Server name *should not* contain underscores. Throughout this paper, 'Repsrv'⁴ is used for this value. As with the in process server, the Reports configurable file for the Reports Server started as an external process. This configurable file resides in the reports/conf directory, with the name <server_name>.conf. It is created when you first start the Reports Server.

The rwservlet.properties configuration file

The rwservlet.properties file is the Reports servlet configuration file located in the reports\conf directory. The servlet properties file contains configuration settings, settings such as those for single sign-on, those for cookie expiry time, and those that determine whether or not an in process server should be started when a server parameter is not contained in the requested URL.

For Forms/Reports integration, you will need to modify the servlet properties file only to disable single sign-on, which is switched on by default.

The <Reports Server>.conf configuration file

The <Reports Server>.conf configuration file is created the first time a Reports Server is started. The <Reports Server>.conf file is an XML file that contains all the information that was stored in the <Reports Server>.ora file in Reports6i, for example, the cache size, the cache directory, the minimum number of Reports runtime engines used, the maximum number of Reports runtime engines used, the job notification settings, and so on.

The cgicmd.dat file

The cgicmd.dat file, located in the reports\conf directory, can be used to shorten the Reports request URL whenever you are either running Reports from the Web or using the Forms WEB.SHOW_DOCUMENT() built-in. The cgicmd.dat file can be used to define keyname/value pairs, where the value defines a number of Reports command-line arguments to be used with one or with many Reports files, each mapped to a named identifier. The following entry in a cgicmd.dat file defines the keyname 'reptest' for the userid, destype, and desformat command-line arguments. The '%*%' indicates that all additional parameters specified in the URL should be added to this command line.

```
reptest: userid=scott/tiger@orcl destype=cache desformat=htmlcss %*%
```

To run a report using the WEB.SHOW_DOCUMENT() built-in in Forms, use this keyname entry:

The complete list of parameters to be used with the cgicmd.dat file is contained in the file itself. Open the file with a text editor to read this information.

⁴ Repsrv is not a unique name, but can be made unique by adding the hostname to it like in 'RepsrvFnmiphui-lap'. Note that there is no price for the shortest Reports Server name.

```
Web.show_document ('/reports/rwservlet?reptest&server=Repsrv&paramform=
no&module=reptest.rdf','_blank');
```

Example 3: Referencing cgicmd.dat keys in Forms calls to WEB.SHOW_DOCUMENT()

Oracle9iAS Reports Services configuration in Oracle9iDS

Oracle9i Reports is configured to use the OC4J integrated HTTP Server to run Reports modules on the Web. This is the default configuration when you install Oracle9iDS. This configuration can also be used to test Reports integration in Forms.

The Reports configuration files explained in this paper are located in the Oracle9iDS installation directory under the reports/conf node.

The Oracle9iDS install does not include the Oracle Single Sign-On Server and Oracle9iAS Portal, which means that you cannot run and test single sign-on integration and Reports access control, provided by Oracle Portal.

To test Forms and Reports integration in a single sign-on environment, you need to have Oracle9iAS installed.

Oracle9iAS Reports Services configuration in Oracle9iAS

When installed with Oracle9iAS, Reports Services uses mod_OC4J and the Oracle HTTP Server to process Web requests. The Reports servlet uses OC4J as its runtime environment.

The Reports software installs into the <Oracle9iAS_Home>\bin directory and the <Oracle9iAS_Home>\reports directory. The Reports Server configuration files are located in the <Oracle9iAS_Home>\reports\conf directory.

Reports Services is single sign-on enabled using mod_osso, an Apache module which is a partner application to the Oracle Single Sign-On Server. Existing Oracle Reports modules can be run in single sign-on mode without any changes to the code being necessary.

SINGLE SIGN-ON INTEGRATION (AVAILABLE WITH ORACLE9iAS)

In the client-server environment, single sign-on support did not apply, and as such Forms and Reports integration did not cover this subject.

Single sign-on is now an integral part of the Oracle9iAS Release 2 offering, and so it can now be used with both Oracle9iAS Forms Services and Oracle9iAS Reports Services, making it an interesting feature to cover for the Forms and Reports integration. Previous versions of Forms and Reports that may have been integrated with the single sign-on capability of Oracle9iAS 1.0.2.x can now be more fully and smoothly integrated.

Oracle9iAS Forms Services and Reports Services are single sign-on enabled through the Oracle9iAS Single Sign-On Server (SSO Server) and the Oracle Internet Directory (OID) for authentication and authorization.

It is important to note that any existing Forms and Reports application can participate in single sign-on with no changes required to the application code. Rather than directly partnering with the Oracle9iAS Single Sign-On Server, the decision to use mod_osso instead was made for backward compatibility.

mod_osso

The HTTP module mod_osso simplifies the authentication process by serving as the sole partner application to the Single Sign-On server, rendering authentication transparent for Oracle9iAS applications in Release2. Oracle9iAS Forms Services and Oracle9iAS Reports Services use mod_osso to register as a partner application to the Oracle Single Sign-On Server

Single Sign-On in Forms

Even when Forms is run in single sign-on mode, the connection to the database is still established with SQL*Net, which requires a physical database account username and password. To support single sign-on, the Forms Servlet is now enabled to look up the user's database account information in the Oracle Internet Directory (OID). Single sign-on in Forms is secure, as there aren't any user credentials sent over the wire to the Forms client. All such exchanges are handled on the middle-tier server.

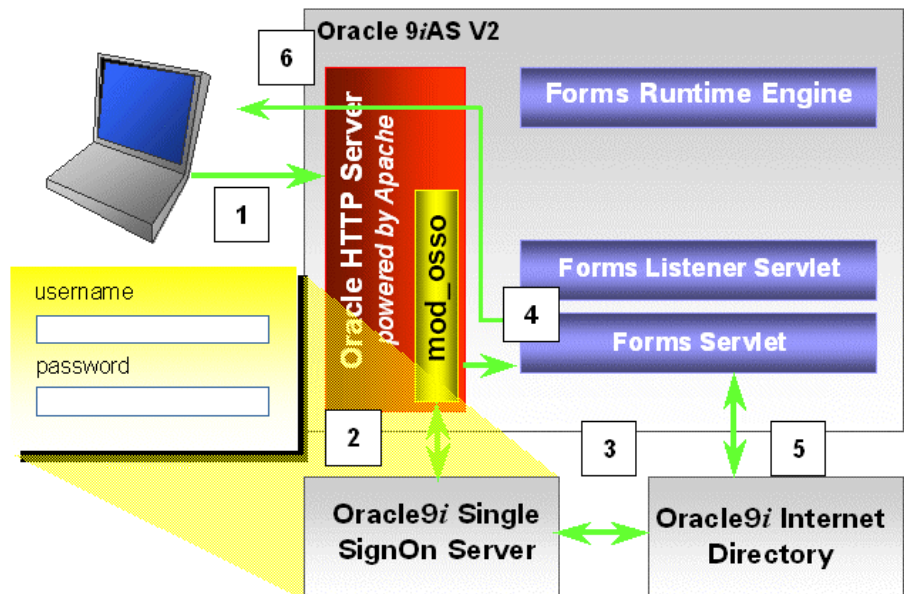


Figure 1: Forms Services Single Sign-On architecture

Single sign-on in Forms proceeds as follows:

1. A user requests a Forms Services application running in single sign-on mode.
Single sign-on mode is activated by registration of the Forms Services URL root “forms90/f90servlet” with mod_osso.⁵ Mod_osso checks to see whether the user is authenticated for this URL, and if not, redirects the request to the Oracle9iAS Single Sign-On Server. If it cannot find a valid authentication cookie within the user request, the Single Sign-On server displays a logon dialog for the user to provide the credentials of his/her lightweight user account
2. Oracle9iAS Single Sign-On Server authenticates the user against the OID.
3. Mod_osso redirects the request to the Forms Servlet, which receives the user single sign-on credentials and then retrieves the physical database user account for the requested application from the Oracle Internet Directory.
4. To obtain the user credentials from the OID, the Forms Servlet looks for the unique key built from the authenticated username and the Forms application name, as it was passed in the “config” parameter of the Forms URL.
5. From this point on, Forms executes as if started in non-ss0 mode.

Enabling single sign-on in Forms

Oracle9iAS Forms Services is installed with single sign-on *disabled*.

To enable single sign-on in Forms, edit the forms90.conf file in the forms90/server directory. Search for the following entries⁶ and remove the comments (“#”) from each.

```
#<IfModule mod_osso.c>
#<Location /forms90/f90servlet>
#require valid-user
#AuthType Basic
#</Location>
#</IfModule>
```

After the mod_osso definition in the forms90.conf file has been uncommented, the single sign-on definition should look like this:

⁵ Single sign-on for Forms is configured in the forms90/server/forms90.conf configuration file.

⁶ In Oracle9iAS 9.0.2 these entries are written into one line. To work with single sign-on these need to be in separate lines like shown in this Whitepaper

```
<IfModule mod_osso.c>

  <Location /forms90/f90servlet>

    require valid-user

    AuthType Basic

  </Location>

</IfModule>
```

Restart the Oracle9iAS HTTP Server to enable these changes. The next time that you request a Forms Web application, you will be prompted for your single sign-on username and password⁷.

Single Sign-On in Reports

Oracle9iAS Reports Services security is a complex subject, covered only briefly in this paper. For more information on Reports security, refer to the whitepaper *Securing Oracle9i Reports*, available at otn.oracle.com.

As in Oracle Forms, the connection to the Reports database connect is established with SQL*Net and a physical database account username and password. Like Forms, Oracle9iAS Reports Services leverages mod_osso for authentication.

The following steps are involved when a user requests a report from a single sign-on protected Reports Server.

Single sign-on in Reports proceeds as follows:

1. A user requests a report from Reports Services with a call to the Reports servlet (<http://.../reports/rwservlet?..>).

For single sign-on to be activated, the SINGLESIGNON parameter in the rwservlet.properties file must be either set to 'Yes' or commented out.

By default Oracle9iAS Reports Services is configured with single sign-on *enabled*. The request is passed from the Reports servlet to mod_osso, which checks with the Oracle9iAS Single Sign-On Server to see whether the user has been authenticated for the Reports URL. If the user has not been previously authenticated, the Single Sign-on Server displays a logon screen for the user to provide his or her single sign-on username and password.

2. The Oracle9iAS Single Sign-On Server uses OID to verify that the provided user credentials are valid.
3. The request is then directed back to mod_osso, which passes the request, in turn, back to the Reports Servlet (rwservlet). Within the Reports URL

⁷ Refer to the Forms Services documentation for directions on setting up Forms Services to serve public requests and single sign-on requests at the same time.

the ssoconn⁸ parameter is used to specify one to many resource(s), each of which is a named identifier in the OID that holds the connection information for a Reports data source. There is no longer any need to pass credential user connect information over the Web using plain text.

4. The Reports Server either uses the same process that the Reports Services servlet runs in, making it an in process server⁹, or runs as an extra process, which is the model also available with Reports 6*i*. The Reports servlet calls the Reports Server, passing the runtime arguments for the report to execute.

Not shown in Figure 2 is the ability of Reports to verify a user's authorization to run a report. The source of authorization is configured by the <security></security> tag pair in the <Reports Server name>.conf file. By default, access control is configured to be performed by Oracle9*i*AS Portal, but it can also use the custom authorization mechanism.

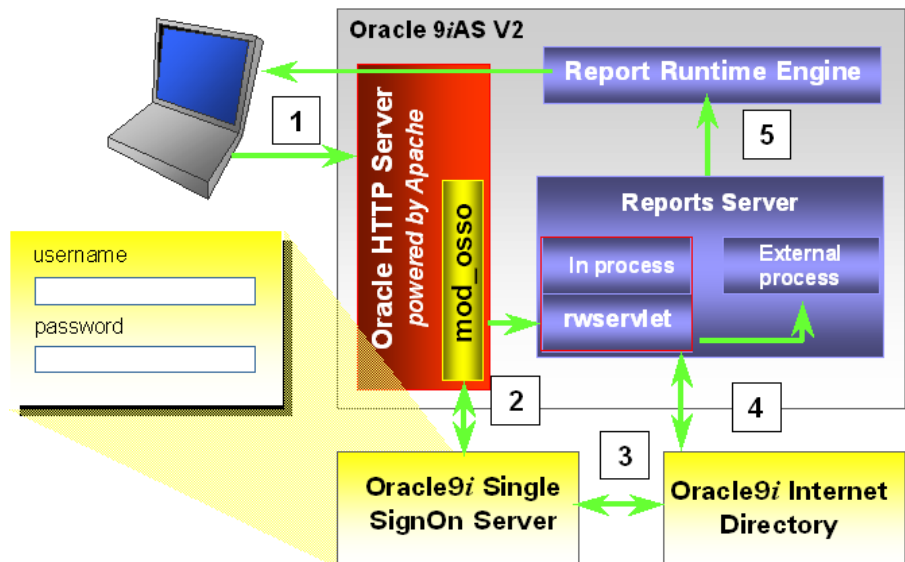


Figure 2: Reports Services Single Sign-On architecture

⁸ SSOCONN can be used when you are calling Oracle Reports using the Forms WEB.SHOW_DOCUMENT() built-in, but not when you are using RUN_REPORT_OBJECT().

⁹ While it is possible to use the in process server with a Reports request initiated from the Forms web.show_document() built-in, you cannot use the in process server with the RUN_REPORT_OBJECT() built-in. The reason for this is that the in process server is implicitly started by the first request to “rwservlet”, which isn’t used with the RUN_REPORT_OBJECT() built-in.

Enabling single sign-on in Reports

Oracle9iAS Reports Services is installed with single sign-on *enabled*.

Enabling access control for Reports Services

Like single sign-on, access control is enabled by default after Oracle9iAS is installed.

Disabling single sign-on in Reports

To *disable* the Reports Server single sign-on, you need to edit the following configuration files:

- reports/conf/rwservlet.properties
- reports/conf/<Reports server name>.conf

In the rwservlet.properties file, search for the SINGLESIGNON parameter, uncomment it, and set the value to No

```
#SINGLESIGNON=YES
```

Change this entry to:

```
SINGLESIGNON=NO
```

Restart the Oracle HTTP Server(OHS) for the changes to take effect.¹⁰

Disabling Oracle Reports access control

To deactivate access control performed by Oracle9iAS Portal, either remove the “securityId=“rwSec”” attribute from the reports/conf/<Reports server name>.conf file ¹¹ or change the <job ../> tag value from

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

to

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

¹⁰ To restart the Oracle HTTP Server in Oracle9iAS, stop the Oracle9iAS Process Manager (OPM), either by using the Service panel (in Windows) or by typing *dmctl stop -ct obs -v -d* on the command line (in Unix). Similarly, to start it up, use *dmctl start -ct obs* (in Unix).

¹¹ Deleting the <security> </security> tag pair is quick and easy, but such deletion has an effect on all <job> types defined in the conf file. This means that access control is disabled for all reports run by this Reports Server. To regain access control, either make a copy of the configuration file before deleting the security entries, so that you can later restore the file, or delete the configuration file and retrieve a new one when restarting the Reports Server.

Restart Reports Services for the changes to take effect.

How does single sign-on relate to Reports integrated in Forms?

As described in this paper, you should perform Reports integration in Oracle Forms on the Web with a server-side call to Oracle9iAS Reports Services, using the RUN_REPORT_OBJECT() built-in in Forms.


When requesting a report from a single sign-on protected Forms Services, the authenticated user's single sign-on identity is implicitly passed to the Reports Server with each call to RUN_REPORT_OBJECT() built-in. The single sign-on identity is used to authenticate the user to the Reports Server for further authorization checking, if required.

A Forms application running in non-sso mode can run a report on a single sign-on secured Reports Server, but fails if the Reports Server requires authorization. Also the enduser must provide his single sign-on credentials when retrieving the Reports output on the Web.

Forms / Reports integration single sign-on matrix

What if Forms Services needs to run in single sign-on mode while Reports Services does not? Or, what if Reports Services runs in access control mode while single sign-on is disabled for both products

The following matrix shows all the possible combination for enabling single sign-on and access control. Access control in this case is performed using the default configuration with Oracle Portal. Also shown is whether a particular combination works with Forms applications that have integrated calls to Oracle Reports.

Oracle9iAS Forms/Reports SSO configuration matrix				
single sign-on mode			Forms built-in	
Forms	Reports		The green checkmark indicates the built-in working with the configuration settings on the left	
SSO	SSO	Access Control	RUN_REPORT_OBJECT	WEB.SHOW_DOCUMENT
NO	NO	NO	✓	✓
NO	NO	YES	No	✓ ¹²
NO	YES	YES	No	✓ ¹³
YES	YES	YES	 ¹⁴	✓ ¹⁵
YES	YES	NO	✓	✓
YES	NO	NO	✓	✓

CALLING ORACLE REPORTS FROM FORMS ON THE WEB

Calling Oracle Reports from Forms on the Web works the same for applications whether they're running in Oracle9iDS or in Oracle9iAS. Everything said about deployment with respect to Forms Reports integration applies to either environment

Introducing the RUN_REPORT_OBJECT built-in

In Oracle9i Forms Developer, to use the RUN_REPORT_OBJECT built-in, you will need to create a new Reports object under the "Reports" node in the Object Navigator. Each Reports object has a logical name which is used within Forms to call the report from PL/SQL. You can create a new Reports object for each physical Reports file. One Reports object can also be used with many physical Reports files

¹² A logon dialog is displayed requesting the system authentication for this report to run. Reports registered with Oracle Portal can be run only by users granted access to it

¹³ The single sign-on logon screen is displayed if the user has not previously been authenticated e.g through connecting to Oracle9iAS Portal. If the report is access controlled through registration with Portal, then only those users that have access permissions with Portal are allowed to run the report.

¹⁴ Supporting RUN_REPORT_OBJECT() with Reports that are access controlled is a planned feature in a future patch set of Forms9i. All applications that run integrated Reports in Forms on client-server do not use access control for Reports, so that there shouldn't exist any upgrade problems when upgrading to Forms9i or moving from Forms client-server to the Web.

¹⁵ If the report is access controlled by registration with Oracle9iAS Portal then only users that have access permissions are allowed to run the report. In this case Forms reports an error when running the report.

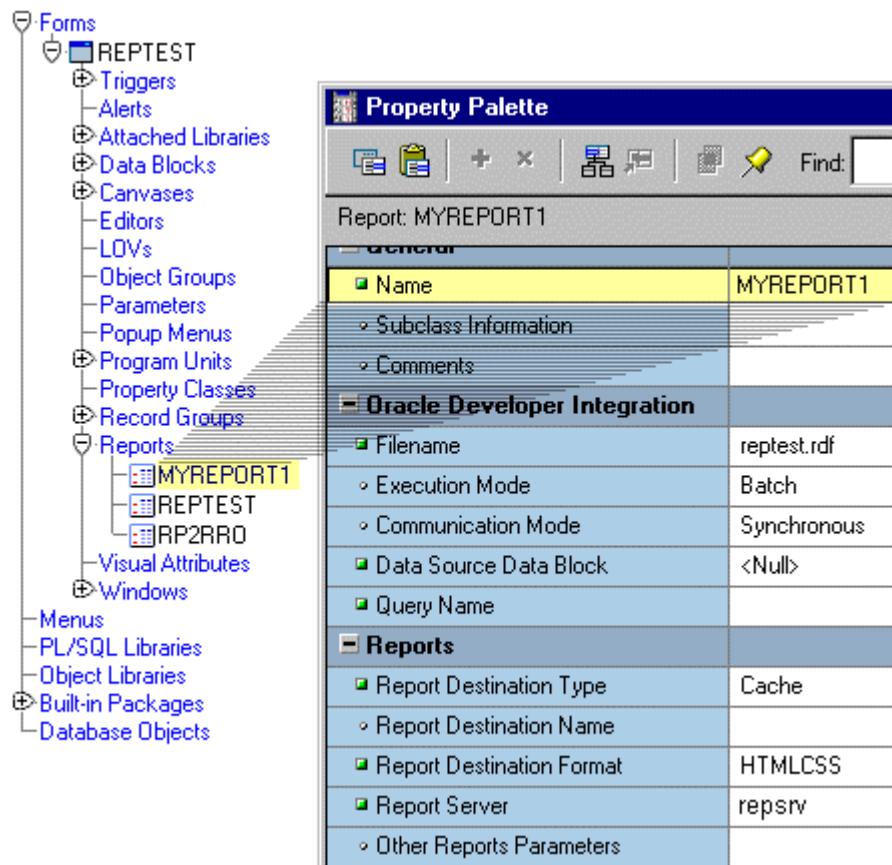


Figure 3: Forms Object Navigator with “Reports” node and Reports objects “MYREPORT1”, “REPEST” and “RP2RRO”. The physical Reports file referenced by the “MYREPORT1” object is defined as “reptest.rdf”. The Reports runtime settings below the “Reports” headline in the property palette can be overwritten during runtime using the `set_report_object_property()` built-in.

General use of RUN_REPORT_OBJECT

The name of the Reports object appears in all caps in the Forms Navigator, but it can be placed in mixed case when called from PL/SQL.

The following example runs a report using the `RUN_REPORT_OBJECT` built-in. The report object node defined in Forms Developer is named “MyReport1”. A user-defined Reports parameter “p_deptno” is passed by Forms using the value in the “dept.deptno” field. The parameter form is suppressed.

```

report_id      Report_Object;
ReportServerJob VARCHAR2(100);

BEGIN
report_id:= find_report_object('MyReport1');

```

```

SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_COMM_MODE,SYNCHRONOUS);
SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESTYPE,CACHE);
SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_SERVER,'Repsrv');
SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_OTHER,'p_deptno=||:Dept.Deptno||
' paramform=no');
ReportServerJob:=run_report_object(report_id);
END;

```

Example 4: General use of RUN_REPORT_OBJECT()

The RUN_REPORT_OBJECT built-in is well documented in the Oracle Forms Services reference manuals and online help.

Using parameterlists in RUN_REPORT_OBJECT

With the RUN_PRODUCT built-in, Reports system parameters and user-defined parameters are passed in a parameter list. The same parameter lists can be used with RUN_REPORT_OBJECT, with the exception of the system parameters, which need to be set by SET_REPORT_OBJECT_PROPERTY() built-in. The following is a list of Reports System parameters to be set when needed.

List of System Parameters in
RUN_REPORT_OBJECT

REPORT_EXECUTION_MODE	BATCH or RUNTIME ¹⁶
REPORT_COMM_MODE	SYNCHRONOUS ASYNCHRONOUS
REPORT_DESTYPE	FILE, PRINTER, MAIL, CACHE ¹⁷
REPORT_FILENAME	The report filename
REPORT_DESNAME	The report destination name
REPORT_DESFORMAT	The report destination format
REPORT_SERVER	The Report Server name

Note that having DESTYPE defined both in the parameter list and in SET_REPORT_OBJECT_PROPERTIES does not prevent the program from compiling, but does prevent it from running.

If your existing parameter list already contains definitions for system parameters, you may experience errors. To prevent problems from arising, modify the

¹⁶ Report_Execution_Mode is a client/server feature and no longer used in Forms9i. Set the value to either BATCH or RUNTIME as it is a required field.

¹⁷ destype 'file' and 'preview' no longer is an option in Oracle9i Reports. If a report needs to be previewed before getting printed then use destype cache with a desformat of htmlcss. If a reports parameter form is required, then issue paramform=yes with the command

parameter list itself, either by removing the entries for DESNAME and DESTYPE, or by adding

```
delete_parameter(<parameter list>,'<name>');
```

to your code before using SET_REPORT_OBJECT_PROPERTIES().

Using the RUN_REPORT_OBJECT built-in¹⁸

The most secure approach for calling Reports from Forms on the Web is to use the Reports Multi-tier Server in combination with RUN_REPORT_OBJECT. Because the user's database connection is implicitly passed from Forms to Reports on the server, there is no risk of interception as there would be if it were passed in the URL. To access a remote Reports Server using RUN_REPORT_OBJECT, the Oracle9i Reports Server must be recognized by the Report Object in Forms. You can do this dynamically, using the SET_REPORT_OBJECT_PROPERTY built-in, or statically, by entering the Reports Server name string into the property palette of the Report Object.

RUN_REPORT_OBJECT example

This example uses a synchronous call to RUN_REPORT_OBJECT to run a Report. It expects the Report object name, the Reports Server name, and the desired output format (PDF, HTML, HTMLCSS) to be passed as a parameter.

```
PROCEDURE RUN_REPORT_OBJECT_PROC (vc_reportobj Varchar2, vc_reportserver
varchar2, vc_runformat varchar2) IS

v_report_id          Report_Object;
vc_ReportServerJob   VARCHAR2(100); /* unique id for each Report request */
vc_rep_status        VARCHAR2(100); /* status of the Report job */
vjob_id              VARCHAR2(100); /* job_id as number only string*

BEGIN

    /* Get a handle to the Report Object itself. */
    v_report_id:= FIND_REPORT_OBJECT(vc_reportobj);

    SET_REPORT_OBJECT_PROPERTY(v_report_id,REPORT_COMM_MODE,
    SYNCHRONOUS);

    SET_REPORT_OBJECT_PROPERTY(v_report_id,REPORT_DESTYPE,CACHE);
```

¹⁸ Note that in Oracle9iDS 9.0.2 there is a known problem using RUN_REPORT_OBJECT() with Forms running in debug mode. This problem is addressed and will get fixed soon. If not running Forms in debug mode then all works fine.

```
/* Define the report output format and the name of the Reports Server as well as a user-defined parameter, passing the department number from Forms to the Report. There's no need for a parameter form to be displayed, so paramform is set to "no". */
```

```
SET_REPORT_OBJECT_PROPERTY(v_report_id,REPORT_DESFORMAT,  
vc_runformat);
```

```
SET_REPORT_OBJECT_PROPERTY(v_report_id,REPORT_SERVER,  
vc_reportserver);
```

```
SET_REPORT_OBJECT_PROPERTY(v_report_id,REPORT_OTHER,  
'p_deptno=||:dept.deptno||'paramform=no');
```

```
vc_ReportServerJob:=RUN_REPORT_OBJECT(report_id);
```

```
vjob_id :=
```

```
substr(vc_ReportServerJob,length(reportserver)+2,length(vc_ReportServerJob)  
);
```

```
/* If finished, check the report status . */
```

```
vc_rep_status := REPORT_OBJECT_STATUS(vc_ReportServerJob);
```

```
IF vc_rep_status='FINISHED' THEN
```

```
/* Call the Reports output to be displayed in a separate browser window. The URL for relative addressing is valid only when the Reports Server resides on the same host as the Forms Server. For accessing a remote Reports, you must use the prefix http://hostname:port/ */
```

```
WEB.SHOW_DOCUMENT ('/reports/rwservlet/getjobid19|| vjob_id  
||'?server='vc_reportserver,'_blank');
```

```
ELSE
```

```
message ('Report failed with error message '||vc_rep_status);
```

If upgrading applications from Forms/Reports 6i to Oracle9i/AS release 2, you will need to modify the Reports job_id, retrieved by the RUN_REPORT_OBJECT() built-in, so that it does not include the Reports Server name when using web.show_document().

¹⁹ The usage of **getjobid** has changed between Reports Server release 6i and Oracle9i Reports Server. In Reports Server 6i the getjobid parameter was used with an equal sign between the parameter name and its value (e.g. getjobid=Repsrv_32). In Oracle9i Reports the equal sign is omitted and the parameter value directly succeeds the name the parameter name (e.g. getjobid32). Also, the job id in getjobid no longer contains the server name.


```
END IF;
END;
```

Example 5: Using RUN_REPORT_OBJECT for integrated calls to Oracle Reports

If you are upgrading applications from Forms/Reports 6i to Oracle9iAS release 2, when calling WEB.SHOW_DOCUMENT() you will need to modify the Reports job_id, retrieved by the RUN_REPORT_OBJECT() built-in, so as not to include the Reports Server name. To use the procedure described above, you would pass the following information in a “When-Button-Pressed Trigger”:

```
RUN_REPORT_OBJECT_PROC(<'REPORT_OBJECT'>,<'REPORT_SERVER_NAME'>',
<FORMAT>)
```

Report_Object	Forms Report object name containing the rdf filename for the Report
Report_Server_Name	Name of the Reports Server
Format	Any of these formats: html htmlcss pdf xml delimited rtf

Forms applications calling a report synchronously make the user wait while the report is processed on the server. For long-running Reports, it is best that you run the report asynchronously, by setting the REPORT_COMM_MODE property to asynchronous and the REPORT_EXECUTION_MODE to batch:

```
SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_EXECUTION_MODE,BATCH);
SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_COMM_MODE,ASYNCHRONOUS);
```

After calling the RUN_REPORT_OBJECT built-in, you must create a timer to run frequent checks on the current Report_Object_Status in a When-Timer-Expired trigger . As a performance recommendation, the timer should not fire more than four times a minute. After the report is generated, the “When-Timer-Expired trigger” calls the WEB.SHOW_DOCUMENT built-in to load the Reports output file, identified by its unique job_id, to the client’s browser.

The following describes the “When-Timer-Expired trigger” that checks for the Report_Object_Status.

```
(...)
/* :global.vc_ReportServerJob needs to be global because the information about the Report
job_id is shared between the trigger code that starts the report and the trigger code (When-
Timer-Expired that checks the current Report status. */
vc_rep_status:= REPORT_OBJECT_STATUS(:global.vc_ReportServerJob);
IF vc_rep_status='FINISHED' THEN
```

```

vjob_id := substr(:global.vc_ReportServerJob,length(reportserver)+2,length
(:global.vc_ReportServerJob));

WEB.SHOW_DOCUMENT ('/reports/rwervlet/getjobid'||: vjob_id
||'?server='vc_reportserver,'_blank');

ELSIF vc_rep_status not in ('RUNNING','OPENING_REPORT','ENQUEUED') THEN
    message (vc_rep_status||' Report output aborted');

END IF;

```

Example 6: Performing asynchronous reporting using RUN_REPORT_OBJECT()

Note: Do not forget to delete the timer when it is no longer needed.

Passing Forms parameter lists in RUN_REPRORT_OBJECT

See Appendix B for a coding example.

The same parameter lists used with RUN_PRODUCT in a client-server environment can also be used with RUN_REPORT_OBJECT when you are calling Reports Services to perform integrated reporting in Forms on the Web. Note that system parameters must be set with the SET_REPORT_OBJECT_PROPERTY built-in. The syntax for using parameter lists in RUN_REPORT_OBJECT is as follows:

```
ReportServerJob:=run_report_object(report_id,paramlist_id);
```

where paramlist_id is the same id used with RUN_PRODUCT²⁰.

Calling Reports that display a parameter form²¹

When you use the RUN_REPORT_OBJECT built-in to create Reports output in Forms, the Reports Server is called directly on the server-side, rather than from the Web. The Reports Server is unaware of the Web access path to the machine that hosts the Reports Server Web interface because this information is not passed with the RUN_REPORT_OBJECT call.

When you use a server-side call to run a report that contains a parameter form, the Reports parameter form in the Web is displayed but is not functional when the user clicks the Submit button. The reason for this can be identified by analyzing the parameter form HTML source code that is generated by Oracle Reports Server:

²⁰ Using RUN_PRODUCT to generate Reports output is not supported in Oracle9i Forms. Forms module containing integrated calls to Reports using RUN_PRODUCT built-in don't compile.

²¹ There exist a known issue with parameter forms in Reports9i when running in single sign-on mode. This problem has been fixed and is contained in a next patch of Reports9i. Until then the code explained in this section does only work for non single sign-on deployments.

(...) <form method=post action=""> (...)

Note that the HTML form "action" tag contains an empty string. For the form to properly generate the final Reports output, a valid action entry is needed, for example:

```
<form method=post action="http://<hostname>/reports /rwservlet?">
```

Similarly, the hidden_run_parameters string value, which is used to store parameter values getting passed implicitly with each subsequent Reports call, also has an empty value:

```
<input name="hidden_run_parameters" type=hidden value="">
```

A valid string for the hidden_run_parameters is as follows:

```
<input name="hidden_run_parameters" type=hidden value="report %3Dreptest
+destype%3Dcache+desformat%3Dhtmlcss+userid%3Dscott%2Ftiger%40fnimphiu+
server% 3DRepsrv">
```

Once you identify the source of the problem, you can fix it within Forms Developer and Reports Developer. Unfortunately, you cannot resolve this problem without modifying the Reports module.

Solution

The solution to the above-mentioned issue includes some programming in both Reports and Forms.

In Forms

To work properly, the parameter form requires three pieces of information that cannot be retrieved in Reports when the parameter is called by RUN_REPORT_OBJECT from Forms:

- username/password@connect_string
- Reports Server name
- Web access path to the Reports Server used.

Consequently, in addition to all other Reports runtime parameters, this information must be passed with the "other" parameter in the RUN_REPORT_OBJECT built-in.

In this example, let's use a procedure in Forms to call RUN_REPORT_OBJECT. The procedure gets a value passed for the report_id, the Reports Server name, and the output format: runformat.

Parameters	Values
report_id	Identifier of the Report Object
reportserver	Name of the Reports Server to use
Runformat	html, htmlcss, pdf, xml ...

Forms code example

```

PROCEDURE RUN_REPORT_OBJECT_PROC(report_id REPORT_OBJECT, reportserver
varchar2, runformat varchar2) IS
ReportServerJob  VARCHAR2(100);
rep_status      VARCHAR2(100);
vjob_id         VARCHAR2(100);
vc_user_name    VARCHAR2(100); /* used for creating parameter form */
vc_user_password VARCHAR2(100); /* used for creating parameter form */
vc_user_connect VARCHAR2(100); /* used for creating parameter form */
vc_connect      VARCHAR2(300); /* used for creating parameter form */

BEGIN
    /* get user connect string */
    vc_user_name:=get_application_property(username);
    vc_user_password:=get_application_property(password);
    vc_user_connect:=get_application_property(connect_string);

    /* creating complete connect string */
    vc_connect:=vc_user_name||'/'||vc_user_password||'@'
||vc_user_connect;

    /*set Reports properties for run_report_object*/
    SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_COMM_MODE,
    SYNCHRONOUS);

```

```

SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESTTYPE,CACHE);

SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESFORMAT,
runformat);

SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_SERVER,
reportserver);

/* P_USER_CONNECT and P_SERVERNAME are custom parameters in the
Reports module */

P_ACTION is also a user-defined variable in Reports and takes the Web access
path to the Report ( it doesn't need to be hard-coded as in this sample ) */
SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_OTHER,'p_deptno=||
:Dept.Deptno||' paramform=yes P_USER_CONNECT=||vc_connect||'
P_SERVERNAME=||reportserver||'P_ACTION=http://fnimphiu-lap'
||'.de.oracle.com:7779/reports/rwservlet?');

report_job_id:=run_report_object(report_id);

rep_status := report_object_status(ReportServerJob);

IF rep_status='FINISHED' then

vjob_id :=substr(ReportServerJob,length(reportserver)+2,length(ReportServerJob));
WEB.SHOW_DOCUMENT('/reports/rwservlet/getjobid||vjob_id||'?
server=Repsrv','_blank');

ELSE

message (rep_status||' Report output aborted');

END IF;

END;

```

Example 7: Forms code to use Reports parameter forms with RUN_REPORT_OBJECT

In Reports

In Reports, the following user-defined variables must be created:

P_ACTIONS	Value for the empty “action” parameter
P_USER_CONNECT	username/password@database used as hidden parameter
P_SERVER_NAME	Reports Server Name

All user parameters are of type character. Don't forget to clear the "Restrict list to predetermined values" check box.

P_ACTIONS has an initial value of _action_.
Omitting this breaks the report when called directly from a Web URL, accessing it via the *rwserver*

Use the following code in a Report "before form" trigger to substitute the default form values (as presented in the Reports module attributes) with your own string.

Reports code example

```
function BeforePForm return boolean is
vc_parameter_form          varchar2(4000);
vc_hidden_runtime_values   varchar2(1000);
vc_report_name             varchar2(100);
begin
    /* If Reports is called from the URL and not from Forms, then p_action is set to its
    default value. In this case, the hidden_value has to retain the default value too */
    If (:p_action='_action_') then
        vc_hidden_runtime_values:='_hidden_';
    else
        /* The Report is started from Run_Report_Object and the hidden parameter has to be
        set */
        /* 1. get the report module name */
        srw.get_report_name(vc_report_name);22

        /* 2. the name needs to be cut off blanks up to the length that it has in characters */
        vc_report_name:=substr(vc_report_name,1,instr (vc_report_name,' ')-1);

        /* Note that I'm not using any custom defined parameters except for
        :p_action,:p_user_connect, :p_servername. If you have additional user-defined
        parameters in your Report output, then this parameter needs to be added to the
        "vc_hidden_runtime_values" string */

        vc_hidden_runtime_values:='report=||vc_report_name||
        '&destype=||:destype||&desformat=||:desformat||'
        &userid=||:p_user_connect||&server=||:p_servername;
    end if;
```

²² The built-in SRW.GET_REPORT_NAME(...) doesn't work properly in Reports 9.0.2. Until this problem is fixed, it is recommended to either hardcode the name of the Reports source file or to pass it as a user parameter within the set_report_object_property (...report_other,...) built-in in Forms

```

/* build the parameter forms HTML code */

vc_parameter_form:= '<html><body bgcolor="#ffffff"><form method=post
action="||:P_ACTION||"><input name="hidden_run_parameters" type=hidden
value="||vc_hidden_runtime_values||"><center><p><table border=0 cellspacing=0
cellpadding=0><tr><td><input type=submit></td><td width=15><input
type=reset></td></tr></table><p><hr><p>';

/* set the modified before form value, to overwrite the default */

srw.set_before_form_html (srw.text_escape, vc_parameter_form);

return (TRUE);

end;

```

Example 8: Reports code to enable Reports parameter forms when called by RUN_REPORT_OBJECT in Forms

The next time you run a Report with an integrated parameter form from RUN_REPORT_OBJECT, the empty values in the HTML source will get replaced. For example:

```

<form method=post
action="http://fnimphiu-lap.de.oracle.com:7779/reports/rwservlet?">

```

and

```

<input name="hidden_run_parameters" type=hidden value="report=Reptest&destype=
Cache&desformat=HTMLCSS& userid=Scott/Tiger@fnimphiu&server=Repsrv">

```

You can copy the HTML template code for building the Report HTML form, as used in the above example, from the Reports “before Form value” property.

Using the WEB.SHOW_DOCUMENT built-in

Use the WEB.SHOW_DOCUMENT built-in procedure to access any Web site from a Forms application on the Web.

Syntax

The following table provides the syntax for WEB.SHOW_DOCUMENT and a brief description of its associated arguments:

```

WEB.SHOW_DOCUMENT ( URL, DESTINATION );

```

URL	The URL is passed as a string (http://www.oracle.com), in a variable, or as a combination of both. If the addressed Web page is located on the same host as the Forms Server, a relative addressing could be used (/virtual_path/page.HTML).
-----	--

DESTINATION Definition of the target where the addressed Web page should be displayed. Values must be single-quoted.

_blank

Displays the Web page in a new browser window.

_parent

Displays the Web page in the parent frame of the current page.

<target_name>

Displays the Web page in a frame specified by the target_name.

A Reports Server is accessible on the Web through the Reports servlet, rwservlet.

```
http://<hostname>:<port>/reports/rwservlet?server=<reportserver_tns>&report=<report>.rdf&desformat=[htmlcss|pdf|xml|delimited]&destype=cache&userid=<user/pw@database>&paramform=[no|yes]
```

Example 9: Calling Reports from a Web URL

The following example calls this Report from Forms on the Web. It assumes that the user parameter “p_deptno” is read from a Forms item “deptno” in the block “dept.”

Example - WEB.SHOW_DOCUMENT ()

```
/* WHEN-BUTTON-PRESSED */  
DECLARE  
vc_url   varchar2(100);  
  
BEGIN  
    vc_url:='http://<hostname><port>/reports/rwservlet?server='  
    ||      'Repsrv&report=reptest.rdf&desformat=htmlcss&destype=cache '
```



```

||      '&userid=user/pw@database&p_deptno=||:dept.deptno||&paramform=no';
WEB.SHOW_DOCUMENT(vc_url,'_blank');

END;

```

Example 10: General use of WEB.SHOW_DOCUMENT()

Example - WEB.SHOW_DOCUMENT() & relative addressing

Use relative addressing if the Reports Server is installed on the same host as the Forms Server.

```

/* WHEN-BUTTON-PRESSED */

DECLARE

vc_url  varchar2(100);

BEGIN

    vc_url:= '/reports/rwservlet?server=Repsrv&report=reptest.rdf&desformat=htmlcss'

    ||

    '&destype=cache&userid=user/pw@database&p_deptno=|| :dept.deptno

    ||

    '&paramform=no';

    WEB.SHOW_DOCUMENT(vc_url,'_blank');

END;

```

Example 11: Using relative Web addresses with WEB.SHOW_DOCUMENT()

Example - Obfuscating the user credentials in the URL

To run Reports securely from Forms in the Web using the WEB.SHOW_DOCUMENT() built-in, it is recommended to run Oracle9iAS Reports Services in single sign-on mode. Refer to the Reports documentation for information on running Reports in single sign-on mode.

Passing the user's name and password to the URL in a human readable format when using WEB.SHOW_DOCUMENT() sometimes makes users feel uncomfortable.

When you call Reports from Forms, you can use a hexadecimal obfuscation of the connect string in the URL that is passed by WEB.SHOW_DOCUMENT(), so that it is not readable on first sight.

This procedure expects the report output format [html, htmlcss, pdf, rtf, xml, delimited] passed as 'runformat' and the name of the Reports definition file.

```

PROCEDURE WEB_SHOW_DOCUMENT_PROC (runformat varchar2,reportname varchar2)
IS
vc_user_name    VARCHAR2(30)    := get_application_property(username);
vc_user_pw      VARCHAR2(30)    := get_application_property(password);
vc_url          VARCHAR2(200);
vc_url_temp     VARCHAR2(300);
v_a            VARCHAR2(10);
v_b            VARCHAR2(10);
i              NUMBER(10);
vc_user_connect VARCHAR2(30):=get_application_property(connect_string);
BEGIN
    /* Create the user's database connect string. */
    vc_url := 'userid=||vc_user_name||/'||vc_user_pw||'@'||
vc_user_connect;
    /* Convert the connect string into a hexadecimal character string. */
    FOR i IN 1..LENGTH(vc_url) LOOP
        v_a := ltrim(to_char(trunc(ascii(substr(vc_url,i,1))/16)));
        if v_a = '10' THEN v_a := 'A';
            elsif v_a = '11' THEN v_a := 'B';
            elsif v_a = '12' THEN v_a := 'C';
            elsif v_a = '13' THEN v_a := 'D';
            elsif v_a = '14' THEN v_a := 'E';
            elsif v_a = '15' THEN v_a := 'F';
        end if;
        v_b := ltrim(to_char(mod(ascii(substr(vc_url,i,1)),16)));
        if v_b = '10' THEN v_b := 'A';
            elsif v_b = '11' THEN v_b := 'B';
            elsif v_b = '12' THEN v_b := 'C';
            elsif v_b = '13' THEN v_b := 'D';
            elsif v_b = '14' THEN v_b := 'E';
            elsif v_b = '15' THEN v_b := 'F';
    END LOOP;
END;

```

```

end if;

vc_url_temp := vc_url_temp||'%'||v_a||v_b;

END LOOP;

/* Create the Reports URL. */

vc_url:='/reports/rwservlet?server=Repsrv+report=||reportname||'+destype=Cache+
desformat=||runformat||'+ '||vc_url_temp ||'+p_deptno=||:dept.deptno;

/* Call the Report in a new browser window using WEB.SHOW_DOCUMENT(). */

WEB.SHOW_DOCUMENT(vc_url, '_blank');

END;

```

Example 12: Hexadecimal URL obfuscation used with WEB.SHOW_DOCUMENT()

Note that in the example above, the Reports Server name is hard-coded to 'Repsrv', which can be avoided by using a parameter in Forms. Also the URL is relative, meaning that the Reports Services installation is on the same Oracle9iAS installation.

Example – Reports SSO and WEB.SHOW_DOCUMENT()

With Oracle9iAS Reports Services and Oracle9iAS Forms Services there exists the option to securely store the application connect information of a user in the Oracle Internet Directory (OID). The connect information is accessed from a named identifier specified with the Forms or Reports URL.

Please refer to the Forms and Reports documentation on how to create resources and assign them to a single sign-on user account.

For example: assume that the database account information is scott/tiger@orcl and that this information was registered in the OID under the named identifier 'myApp'. To use this datasource from Forms and Reports running in sso mode, all you need to do is to specify the name 'myApp' in the Forms 'config' URL parameter and in the Reports 'ssoconn' URL parameter when requesting the application. The combination of the single sign-on username and the named identifier is unique for each user, thus allowing the Forms and Reports servlets to find the user's connect information in the OID during startup. The following call to WEB.SHOW_DOCUMENT() is assumed to come from a Forms application that was started in sso mode. This time, the single sign-on user is authenticated and the user name is known in the browser session.

```

/* WHEN-BUTTON-PRESSED */
DECLARE
vc_url    varchar2(100);

BEGIN
    vc_url:= '/reports/rwservlet?server=Repsrv&report=reptest.rdf&desformat=htmlcss'
        || '&destype=cache&ssoconn=myApp&p_deptno='|| :dept.deptno ||
        '&paramform=no';
    WEB.SHOW_DOCUMENT(vc_url,'_blank');
END;

```

Example 13: Using Reports single sign-on authentication with WEB.SHOW_DOCUMENT()

The Reports Servlet uses the value specified in the ssoconn parameter to retrieve the datasource connect information stored in the OID for the connected user.

If the user has not yet been authenticated (as would happen, for example, when Forms is not running in single sign-on mode) before the report will execute, a single sign-on logon dialog is displayed for the user to authenticate..

Printing Reports output on the Web

Reports output created with the Reports Server is sent to a printer using the destype=printer and desname= \\network_access\printer_name arguments. Because the Reports output is created on the middle-tier server, there is no local printer support. To send a report to a local , make sure that the printer is configured as a network printer on the middle-tier server which hosts the Reports Server.

FORMS MIGRATION ASSISTANT (FMA)²³

The Forms Migration Assistant (FMA) provided with Oracle9i Forms contains a utility to migrate integrated calls to Oracle Reports which originally used the RUN_PRODUCT built-in to use RUN_REPORT_OBJECT instead. The FMA is an automated tool, and as such it cannot provide the same flexibility that RUN_REPORT_OBJECT provides in a manual migration. However, if you don't know how to use RUN_REPORT_OBJECT in Forms or if your business doesn't allow longer maintenance intervals when migrating your Forms application to the

²³ Oracle recommends using the FMA that comes with the first patch set of Forms9i. All problem described in this section are fixed within the patch.

Web, the FMA gives you a working implementation of the RUN_REPORT_OBJECT²⁴ built-in.

Known issue with rp2rro.pll and workaround

There exists a known issue with the RUN_PRODUCT() to RUN_REPORT_OBJECT() migration utility in the FMA version shipped with the base release of Oracle9iAS Release2. The problem is in the Forms library rp2rro.pll, which handles converted RUN_PRODUCT() calls after the migration. To fix this problem, open the rp2rro.pll file in Oracle9i Forms Developer and double-click the package body.

Search for the string “getjobid=” to find the following code segment :

```
WEB.SHOW_DOCUMENT(rp2rroVirtualDir||rp2rroReportsInterface||'/getjobid='  
||rp2rro_jobidFull||'?server='||rp2rroReportServer,'_blank');
```

Change this code line to

```
WEB.SHOW_DOCUMENT(rp2rroVirtualDir||rp2rroReportsInterface||'/getjobid'  
||rp2rro_jobidPartial||'?server='||rp2rroReportServer,'_blank');
```

Without this change, the Reports Server displays an error message showing a numeric format exception.

If you don't have a database account accessible with the rw_server_queue table installed, then to compile rp2rro.pll, in the same package search for “rp2rro_getQueueTableErrors”.

Edit the rp2rro_getQueueTableErrors function with the elements highlighted in bold.

²⁴ Please refer to the documentation of the Forms Migration Assistant about how to use it and what to expect from the RUN_REPORT_OBJECT migration.

```

FUNCTION rp2rro_getQueueTableErrors (jobID varchar2) RETURN Varchar2 IS

vErrors varchar2 (4000):= ' ';

-- CURSOR getErrorsFromQueueTable IS select status_code ||' - '
||status_message from rw_server_queue where job_id = to_number(jobID);

BEGIN

-- open getErrorsFromQueueTable;

-- fetch getErrorsFromQueueTable into vErrors;

-- close getErrorsFromQueueTable;

-- return vErrors;

null;

EXCEPTION

WHEN OTHERS THEN

return 'rp2rro error: Reports Queue table not available';

END;

```

Example 14: Disabling the requirement of the rw_server_queue table install to recompile rp2rro.pll

You should now be able to compile the rp2rro.pll file and run your converted Reports. This problem, as mentioned earlier, is a known problem and will be fixed in a future Oracle9i Forms patch sets.

SUMMARY

This paper highlighted the options available for integrating calls to Oracle9i Reports in Oracle9iAS Forms Services on the Web. The RUN_PRODUCT() built-in is no longer available in Forms to call out to Oracle Reports. Instead, either the RUN_REPORT_OBJECT() built-in, first introduced in Oracle Forms 5.0, or the WEB.SHOW_DOCUMENT() must be used for this task.

New in Oracle9iAS Forms and Reports is single sign-on integration. In Forms and Reports, single sign-on is performed with mod_osso in the Oracle HTTP Server and the Oracle9iAS Single Sign-On Server. For Reports integration in Forms to work, it is recommended that you run Forms and Reports in the same mode, single sign-on or non single sign-on.

Appendix: Usability matrix

This function matrix compares the options described in this paper for running Reports from Forms on the Web.

Integrating Oracle9i Reports in Oracle9iAS Forms Services		
Functionality	RUN_REPORT_ OBJECT	WEB.SHOW_ DOCUMENT
Multiuser-enabled	✓	✓
Parallel Reports processing	✓	✓
Scalable	✓	✓
Asynchronous reporting	✓	
Reports parameter form support	✓ ²⁵	✓
Using Forms parameter lists	✓	
Secure user name/ password	✓	✓ ²⁶
Report status notification	✓	
User-defined parameters	✓	✓
Running Reports on a host other than the Forms Services	✓	✓
Multiple output formats	✓	✓
Recommended	✓	✓

Appendix B: Passing Forms parameter lists

Passing a parameter to a report with RUN_REPORT_OBJECT differs from passing it with RUN_PRODUCT. However, it is still possible to use the parameter lists created to run with RUN_PRODUCT in combination with RUN_REPORT_OBJECT.

The following code includes a parameter list in a call to RUN_REPORT_OBJECT that submits a Report to the Reports Multi-tier Server.

²⁵ Requires additional coding as explained in this paper

²⁶ Only with single sign-on

Forms Parameter list example

```
PROCEDURE RUN_REPORT_OBJECT_LIST(report_id REPORT_OBJECT, reportsserver
varchar2,
runformat varchar2) IS
ReportServerJob VARCHAR2(100);
vc_rep_status VARCHAR2(20);
paramlist_id ParamList;
paramlist_name VARCHAR2(10):='tmplist';
BEGIN
    /* The Reports_Object Properties needs to be set for using the Reports Server .*/
    SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_COMM_MODE,
    SYNCHRONOUS);
    SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_SERVER, reportsserver);
    /* Check for and delete parameterlist. */
    paramlist_id:= get_parameter_list(paramlist_name);

    IF NOT id_null(paramlist_id) THEN
        destroy_parameter_list(paramlist_id);
    END IF;

    paramlist_id:=create_parameter_list(paramlist_name);

    /* The parameterlist determines the destype, the Reports output format, and the user
    defined variable read from a Forms file's :dept.deptno. */

    add_parameter(paramlist_id,'DESTYPE',TEXT_PARAMETER,'CACHE');
    add_parameter(paramlist_id,'PARAMFORM',TEXT_PARAMETER,'NO');
    add_parameter(paramlist_id,'p_DEPTNO',TEXT_PARAMETER,
    :DEPT.DEPTNO);
    add_parameter(paramlist_id,'desformat',TEXT_PARAMETER, runformat );
    ReportServerJob:=RUN_REPORT_OBJECT(report_id,paramlist_id);
END;
```

Example 15: Using Forms parameter lists with RUN_REPORT_OBJECT()

Important: If your existing parameter list already contains definitions for system parameters, this list will overwrite the configuration in SET_REPORT_OBJECT_PROPERTY(). To avoid confusion or unwanted behavior, we recommend that you modify the parameter list itself, removing the entries for DESNAME and DESTYPE, or that you add

```
delete_parameter(<parameter list>,'<name>');
```

to your code before using SET_REPORT_OBJECT_PROPERTIES().

Note that having DESTYPE defined both in the parameter list and in SET_REPORT_OBJECT_PROPERTIES does not prevent the program from compiling, but will prevent it from running.

Appendix C: Troubleshooting

The following table lists some of the problems that can arise when you work with Forms on the Web that have integrated calls to Oracle Reports.

Problem	Solution
<p>The Reports Server started with <code>rwserver server=<server_name> batch=yes</code> starts up but is not accessible from the Reports servlet Web interface.</p>	<p>Reports Servers are CORBA based and check the network to see whether a started Reports Server name already exists. If a Reports Server with the same name exists, the second Reports Server shuts itself down. On Windows, you can see an error message if you are not using <code>batch=yes</code> when starting the Reports Server process. To be on the safe side, always specify the host name within the name of the Reports Server: 'RepsrvFnimphiu-lap' is less ambiguous than 'Repsrv'</p>
<p>After an upgrade from Forms6i to Oracle9i Forms, calls to <code>RUN_REPORT_OBJECT()</code> create a Reports output on the server, but the output cannot be accessed using <code>getjobid</code>.</p>	<p>In Oracle9iAS Reports Services, the format of the value used with 'getjobid' has changed.</p> <p>In Reports6i, the format was <code>getjobid=<report_server_name>_<job number></code></p> <p>In Oracle9i Reports this has become <code>Getjobid<job number></code></p> <p>Add the following line to the trigger including the <code>RUN_REPORT_OBJECT()</code> call to fix</p>

	<p>the problem:</p> <pre>vjob_id := substr(report_message, length(reportserver)+2,length (report_message));</pre> <p>Use "...getjobid" vjob_id "..." in your call to WEB.SHOW_DOCUMENT()</p>
<p>Oracle9iAS Forms Services cannot run integrated Reports. The job fails with an error message saying that no job status could be obtained.</p>	<p>The Reports Server is configured incorrectly for access control. Open the Oracle9iAS_Home/reports/conf/<Report_Server_Name>.conf configuration file and remove the <security>...</security> tag pair along with its content.</p>
<p>Using RUN_REPORT_OBJECT() in Forms does not work with Reports in process server.</p>	<p>Forms requires the Reports Server to run in an extra process. This is the case because the call to Oracle Reports is performed on the server-side and not via the Reports Servlet. If you want to use the in process server, you must start the Reports integration using the WEB.SHOW_DOCUMENT() built-in.</p>



Integrating Oracle9iAS Reports Services in Oracle9iAS Forms Services
December 2002

Author: Frank Nimphius

Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2002 Oracle Corporation
All rights reserved.