Oracle Application Server 10*g* - Integrating Oracle Reports in Oracle Forms Services applications

An Oracle White Paper
May 2004 (updated)

ORACLE

Oracle Application Server 10*g* -Integrating Oracle Reports in Oracle Forms Services applications

Oracle Application Server 10*g* -Integrating Oracle Reports in Oracle Forms Services applications

INTRODUCTION

Oracle Application Server 10*g* (9.0.4) and Oracle Developer Suite 10*g* (9.0.4) contain Oracle Forms 9.0.4 and Oracle Reports 9.0.4.

This paper discusses *how* to call Oracle Reports that are integrated in Oracle Forms applications. 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 how to integrate Oracle Reports in Oracle Forms Services when running in Single Sign-On mode

The audiences that are addressed in this paper are customers and technical consultants who know the Forms and Reports integration in client/ server environments and who plan to move their applications to the Web using Oracle Forms and Oracle Reports in Oracle Application Server 10*g* and Oracle Developer Suite 10*g*.

ORACLE FORMS

Oracle Forms consists of two components: the Oracle Forms Developer building tool (commonly known as the Builder) and the Oracle Application Server 10*g* Forms Services Web deployment component.

There are two Oracle 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.

Oracle Forms Services is a component of the Oracle Application Server 10*g* (OracleAs 10*g*) for production deployments and also part of the Oracle Developer

Suite 10*g* for testing. The Forms Services components and configuration files provided in the Oracle Developer Suite 10*g* are the same as those in OracleAS 10*g*, so that an application developed with 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 Oracle Developer Suite 10*g* or Oracle Application Server 10*g* 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:

- 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. For example:

```
[reptest]
form=reptest
userid=scott/tiger@orcl
look&feel=oracle
```

```
width=700
height=500
```

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

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

For Oracle Forms and Oracle Reports integration to work, you do not need to specify the Reports_Path variable in the Forms default.env file. The Reports application modules are accessed by Oracle Application Server 10*g* Reports Services and must be accessible for this component.

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, 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. Next, 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 it is 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/ l90servlet".

OC4J and mod_oc4j

Oracle Containers for J2EE (OC4J), the default servlet container in Oracle Application Server 10*g* and Oracle Developer Suite 10*g*, is the runtime environment for the Forms Servlet and the Forms Listener Servlet component. mod_oc4j routes requests from the Oracle HTTP Server to Oracle Application Server Containers for J2EE (OC4J), through the AJP 1.3 protocol.

Forms Services configuration in Oracle Developer Suite 10*g*

To conserve disk space, Oracle Developer Suite 10*g* 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 Oracle Application Server 10*g*, except that neither mod_oc4 nor Infrastructure services such as Single Sign-On is available in Oracle Developer Suite 10*g*[1].

The Forms Web runtime environment is automatically configured when Oracle Application Server 10*g* is installed with the Forms and Reports option. No

[1] With Oracle Application Server 10*g* (9.0.4) it is possible to install Oracle Forms and Oracle Reports only. However, Infrastructure services such as Single Sign-On, are not available..

additional manual configuration is required to call Reports from Forms applications.

Forms Services configuration in OracleAS 10*g*

The Forms Servlet and the Forms Listener Servlet that are running in OC4J are accessed through mod_oc4j, which is an Oracle Apache module. When your application is running in OracleAS 10*g*, mod_oc4 routes all calls to / forms90/ f90servlet and / forms90/ l90servlet to OC4J, while the Oracle HTTP Server handles all the downloads of images and Java archive files that are used within a Forms application.

All required configurations are automatically performed when Oracle Application Server is installed. As with Oracle Developer Suite 10*g*, there is no need for additional manual configuration to call Reports from Forms applications.

ORACLE REPORTS

Like Forms, Oracle Reports is the brand name for two separate components: the Oracle Reports Developer building environment and the Oracle Application Server 10*g* Reports Services deployment environment. Throughout this paper, the terms Reports Services and Reports Server are used interchangeably for the same component. Oracle Reports Developer is part of the Oracle Developer Suite 10*g* offering and is available on the same platforms as is Oracle Developer Suite.

Oracle Application Server 10*g* Reports Services, the Reports Server runtime component, is included in Oracle Application Server 10*g*.

Oracle Reports Services, the Web deployment environment for Oracle Reports modules, is part of Oracle Application Server 10*g*. Oracle 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 Server process
- Rwservlet.properties configuration file
- <Reports Server>.conf configuration file
- The cgicmd.dat file

The Reports Servlet

The default Oracle Application Server 10*g* Reports Services Web interface is a servlet, *rwservlet,* which dispatches incoming Reports HTTP requests to the Reports Server engine and performs Single Sign-On authentication where required.

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

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 Oracle Reports Services: the Reports 'in process' server and the Reports server running in an extra process.

The Reports 'in process'[2] server is started through the Reports Servlet *rwservlet* 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 *rwservlet*, 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. The Reports Server can be started by the following command, accessible from the \ bin directory of the Oracle Developer Suite or Oracle Application Server 10g installation.

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

or

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

Any value can be used for <server_name>, as long as the name selected is unique in the accessible network. Throughout this paper, 'Repsrv[3]' is used for this value.

The Reports configuration file for the Reports Services 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 andReports integration, you will need to modify the servlet properties file only to disable Single Sign-On, which is switched on by default.

---

[2] 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.
[3] Repsrv is not a unique name, but can be made unique by adding the hostname to it like in 'Repsrv@fnimphiu-pc'. Note that there is no added benefit for the shortest Reports Server name.

The <Reports Server>.conf configuration file

The <Reports Server>.conf configuration file is created the first time a Reports Server is started.

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.

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.

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

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

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

Reports Services configuration in Oracle Developer Suite 10*g*

Oracle Reports 10*g* (9.0.4) is configured to use the OC4J integrated HTTP Server to run Reports modules on the Web. This is the default configuration when you install Oracle Developer Suite 10*g*. This configuration can also be used to test Reports integration in Forms.

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

Oracle Developer Suite does not include the Oracle Single Sign-On Server and Oracle Application Server 10*g* 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 Oracle Application Server 10*g* installed.

Reports Services configuration in OracleAS 10*g*

When installed with Oracle Application Server 10*g*, 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 <OracleAS_Home>/ bin directory and the <OracleAS_Home>/ reports directory. The Reports Server configuration files are located in the <OracleAS_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 making any code changes.

THE RUN_REPORT_OBJECT () BUILT-IN

The most secure approach for calling Reports from Forms on the Web is to use the Oracle Application Server 10g reports Services 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 if it were passed in a URL.

In Oracle 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

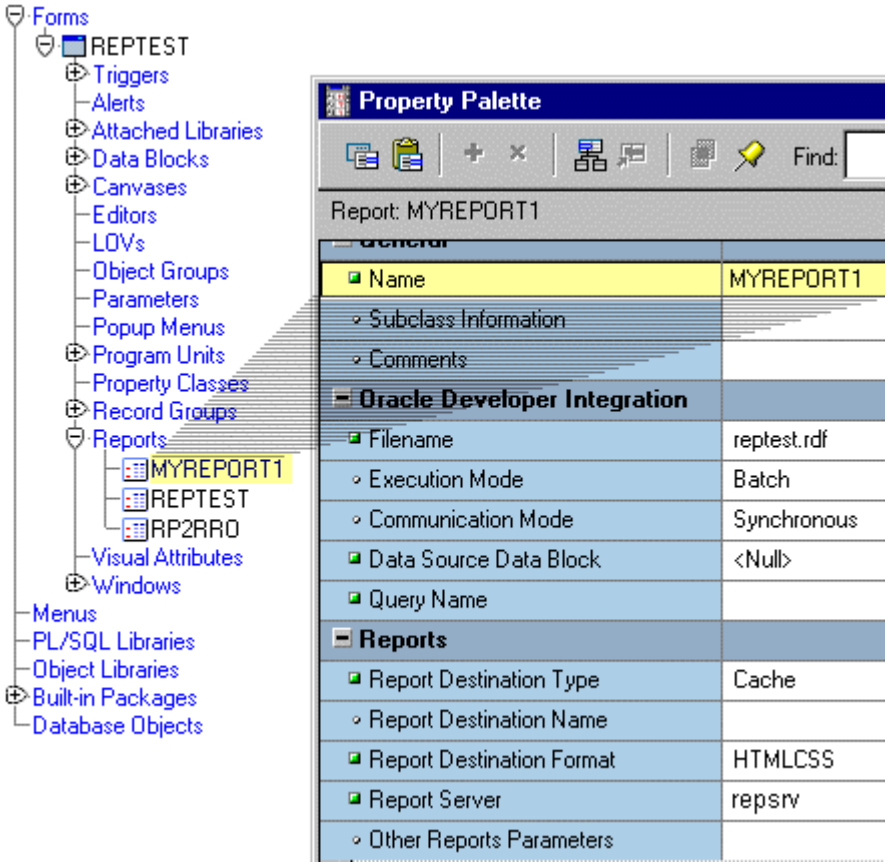

Figure 1: Forms Object Navigator with "Reports" node and Reports objects "MYREPORT1", "REPTEST" and "RP2RRO". The physical Reports file referenced by the "MYREPORT1" object is defined as "reptest.rdf". The Reports runtime settings below the "Reports" headline in the property palette

can be overwritten during runtime using the set_report_object_property() Built-in.

How to use Run_Report_Object

To access a remote Reports Server using RUN_REPORT_OBJECT, Oracle Application Server 10*g* Reports Services must be accessible for 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.

The following example runs a report using the Forms Developer Built-in RUN_REPORT_OBJECT. The Report Object name is "MyReport1". A user-defined Reports parameter, "p_deptno", is passed using the value of the "dept.deptno" field. The parameter form is suppressed using "paramform=no".

```
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.De
ptno||' paramform=no');

ReportServerJob:=run_report_object(report_id);

END;
```

Example 1: General use of RUN_REPORT_OBJECT()

Run_Report_Object examples

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_reportoj 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_reportoj);
       SET_REPORT_OBJECT_PROPERTY(v_report_id,REPORT_COMM_MODE,
       SYNCHRONOUS);
       SET_REPORT_OBJECT_PROPERTY(v_report_id,REPORT_DESTYPE,CACHE);


/* 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_Repo
rtServerJob)
       );
/* 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/getjobid[4]'||  vjob_id
        ||'?server='vc_reportserver,'_blank');


        ELSE

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

        END IF;

END;
```

Example 2: Using Run_Report_Object for integrated calls to Oracle Reports

If you are upgrading applications from Forms or Reports 6*i* to Oracle Application Server 10*g*, when calling WEB.SHOW_DOCUMENT() you will need to modify the Reports job_ID that is retrieved by the RUN_REPORT_OBJECT() Built-in, so that the Reports Server name is not included.

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);
```

[4] The usage of **getjobid** has changed between Reports Server release 6*i* and Oracle9*i* Reports Server. In Reports Server 6*i* the `getjobid` parameter was used with an equal sign between the parameter name and it's value (e.g. getjobid=Repsrv_32). In Oracle9*i* 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.

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. 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/rwservlet/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 3: Performing aynchronous reporting using RUN_REPORT_OBJECT()

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

Using parameter lists with 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 with the 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 [5]
REPORT_COMM_MODE	SYNCHRONOUS ASYNCHRONOUS
REPORT_DESTYPE	FILE, PRINTER, MAIL, CACHE[6]
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 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( ).

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_PRODUCT`[7].

Calling Reports that display a parameter form

Using the Forms RUN_REPORT_OBJECT() Built-in to call Oracle Reports that contain a parameter form requires code changes in Forms to run on the Web.

Please read the Whitepaper "Oracle Forms Services – Using Run_Report_Object() to call Reports with a parameter form" available on http:/ / otn.oracle.com / products/ forms/ .

---

[5] Report_Execution_Mode is a client/ server feature and no longer used in Oracle Forms 10*g* (9.0.4.) Set the value to either BATCH or RUNTIME as it is a required field.

[6] destype 'file' and 'preview' no longer is an option in Oracle 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

[7] Using RUN_PRODUCT to generate Reports output is not supported in Oracle Forms 10*g* (9.0.4). Forms module containing integrated calls to Reports using RUN_PRODUCT Built-in won't compile.

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 4: 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 5: 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 6: Using relative Web addresses with WEB.SHOW_DOCUMENT()

Hiding username and password

To execute a report on the Web, the database connect information must be passed as part of the request URL if you're not using Single Sign-On in Oracle Application Server 10*g*. Adding sensitive user information to any URL request is a serious security breach because all URLs requested by a user can be looked up in the Browser's URL history.

If Single Sign-On is not an option to implement and Reports must be run through the Web.Show_Document() Built-in , then, to avoid exposing the database connect information in the Reports request URL, it is possible to store the username and password pair in a temporary cookie on the client that can be read by the Oracle Reports Servlet. The cookie is set by a Java Bean on the Forms Java client and expires immediately after the Browser Window is closed.

Please read the Whitepaper "Oracle Forms Services - Secure Web.Show_ Document calls to Oracle Reports", available at http:/ / otn.oracle.com/ products/ forms/ for a detailed description and coding example.

PRINTING REPORTS ON THE WEB

Unlike in client-server where local printers were used by Oracle Reports printing, a network printer configured on the server that hosts Oracle Application Services 10g Reports Services, is required.

Another option is to download the Reports output in a printable format, like PDF, to the client Browser and print it from a plugin or local client software.

FORMS AND REPORTS INTEGRATION IN SINGLE SIGN-ON MODE[8]

Oracle Application Server 10*g* Forms Services and Oracle Application Server 10*g* Reports Services can run in Single Sign-On mode without any changes required to the Forms and Reports application modules.

Using Oracle Single Sign-On, all user information and credentials are stored securely in Oracle Internet Directory (OID), a LDAP v3 compliant directory server.

Oracle Application Server 10*g* (9.0.4) Reports Services by default installs with Single Sign-On enabled[9]. To change the Oracle Reports installation to not use Single Sign-On, uncomment the Single Sign-On parameter in the rwservlet.properties file, which is located in the <Oracle Application Server Midtier Home>/ Reports/ conf directory, and set its value to false:

---

[8] A known limitation of Forms and Reports running in Single Sign-On mode is that the Single Sign-On username – also known as the common name (cn) - defined in OID must not contain blanks. This limitation will be addressed in a patch set.

[9] Note that you need to install Oracle Application Server 10*g* Enterprise Edition for using Single Sign-On. The Forms and Reports stand alone installation option – shipped as a separate CD of Oracle Application Server 10*g* –cannot use Single Sign-On.

```
SINGLESIGNON=NO
```

After restarting Oracle Application Server 10*g*, Reports no longer requires Single Sign-On authentication.

Oracle Application Server 10*g* Reports Services requires a system authentication to verify the users right to access a Reports source file. Access control is, like Single Sign-On, enabled by default and verifies the users access privileges against Oracle Portal[10].

Even if the Reports source files are not access protected, at least a valid Portal username/ password pair must be provided for authentication. This can be done in a Reports authentication dialog or by adding the "authid" parameter to the Reports application request. The authid parameter value must be a valid username/ password pair.

Using Run_Report_Object() to call Oracle Reports that are access protected from Forms, the authid parameter needs to be added to the call to

```
Set_Report_Object_Property(rep_id, REPORT_OTHER,'..');
```

To disable access control[11], open the Reports Services configuration file that is located in the <Oracle Application Server Midtier Home>/ Reports/ conf directory. The Reports Services configuration file has the name of the Reports Services and the file extension '.conf'.

For example:

```
repserv10gas@fnimphiu-pc.conf
```

In the Reports configuration file, remove the <security> </ security> tag pair with all the content it encloses. Restart the Reports Services for the changes to apply.

To bring back access control, delete the Reports Services configuration file and start the Reports Server. In the absence of the Reports Services configuration file, a new file is created.

For a complete overview of single sign-on in Oracle Forms 10g, please read the Whitepaper "Oracle Application Server 10*g* (9.0.4) - Forms Single Sign-On" available from http://otn.oracle.com/products/forms.

To enable Single Sign-On in Forms, as it is disabled after installing Oracle Application Server 10*g* Forms Services, open the formsweb.cfg[12] file that is located in the <Oracle Application Server Midtier Home>/ Forms90/ server directory and add ssoMode=true to the named configuration of the application that should run in Single Sign-On mode.

---

[10] If not requiring access control for Reports application modules it is recommeded to disable access control when calling Oracle Reports from Forms

[11] Forms and Reports applications that are upgraded from Oracle Forms and Reports version 6*i* normally don't use access control to protect reports sources. In this case it is recommended to disable Single Sign-On and access control setting.

[12] The formsweb.cfg file can be edited more safely using Oracle Enterprise Manager Application Control. The Forms configuration page provides a UI to perform changes to the formsweb.cfg file.

Using the Run_Report_Object () Built-in

Using the Run_Report_Object() Built-in for calling Oracle Reports from Forms, no modification is required for enabling Single Sign-On. A Forms application that runs in Single Sign-On mode passes the authenticated SSO username to Reports internally using the Reports 'authid' parameter. Because Forms is trusted by Reports, no password needs to be passed.

Using the Web.Show_Document() Built-in

Using Web.Show_Document() for calling Oracle Reports from Forms running in Single Sign-On mode requires a Resource Access Descriptor (RAD) to be created in OID.

A Resource Access Descriptor is a credential store for the users database username and password for this particular application. A RAD can be defined globally for all users in OID, or individuallyper user.

The name of the RAD must be contained in the Reports Services request URL. The parameter name for this information is 'ssoconn'.

```
http://fnimphiu-
pc.us.oracle.com/reports/rwservlet?report=mySaveReport&ssoconn=myApp&….
```

The Reports Servlet reads the authenticated user's database credentials from OID, where the value of ssoconn, in combination with the user's SSO name, builds a unique key for querying OID user database credentials.

SUMMARY

Forms and Reports on the We integrate either using the Run_Report_Object() Built-in or the Forms Web.Show_Document() Built-in. Use the Run_Report_ Object() Built-in to securely pass reports parameters to the Oracle Application Server Reports Services. Web.Show_Document() is a good choice if the in process Reports Server should be used. Further more, Web.Show_Document() is less complex to code.

Forms and Reports integrate with the Oracle Single Sign-On Server. A user, once authenticated to Oracle Application Server Forms Services running in SSO mode, is automatically authenticated to run Oracle Reports Services.

APPENDIX: RUN_REPORT_OBJECT - COMPLETE EXAMPLE

The PL/ SQL example shown below can be used generically to call Oracle Reports form from Oracle Application Server Forms Services. This example handles the cases where Forms and Reports run in Single Sign-On mode and non-Single Sign-On mode.

To integrate this example code in your Forms applications, two user parameters need to be created in Forms: REPORTSSERVER and CONFIG.

Add the following entry to the application's named configuration in the formsweb.cfg configuration file. The formsweb.cfg file is located in the forms90/ server directory.

```
[reptest]

form=reptest.fmx

…

otherparams=REPORTSSERVER=repserv10gAs@fnimphiu-pc CONFIG=%CONFIG%

…
```

The "otherparams" parameter provides the Forms application with the name of the Reports Server used as well as the name of the Forms configuration – reptest – which in case Single Sign-On also represents the name of the Resource Access Descriptor (RAD). Resource Access Descriptors act as a credential store in Oracle Internet Directory (OID) and hold Forms and Reports database connect information for individual users and applications.

For a complete overview of single sign-on in Oracle Forms 10g, please read the Whitepaper "Oracle Application Server 10*g* (9.0.4) - Forms Single Sign-On" available from http://otn.oracle.com/products/forms.

The following arguments are required by the RUN_REPORT_OBJECT_PROC PL/SQL procedure shown below:

report_id – The Report Object handle

report_server_name – The name of the Reports Service used to execute the requested report

report_format – The format of the Reports output

report_destype_name – Reports destype on the Web should be Cache but can also be Printer, File or Mail

report_file_name – Name of the physical Reports (".rdf" or ".rep") to execute. The Reports file must be accessible to the Reports Server

report_otherparam – Additional, non-system Reports parameters like paramform= yes| no to enable the Reports parameter form

reports_servlet – The absolute access path to the Reports Servlet (by default / reports/ rwservlet)

The following code, executed in a When-Button-Pressed trigger, calls a Report that requires a parameter form. The Forms Reports Node is assumed to be named 'reptest". The "v_report_other" variable specifies the Reports parameter form to be displayed and passes a custom Reports parameter p_deptno[13] with its value read from a Forms text field. The Reports output format is chosen as HTMLCSS and the destination type is CACHE, which means the Reports output is displayed in the Browser of the user requesting the report. The physical Reports source file to be executed is "reptest.rdf". The Oracle Application Server 10g Reports Services is assumed to be installed on the same physical server machine, which is why the reference to the Reports Servlet is not using absolute addressing, but a relative address.

```
DECLARE
        report_id Report_Object;
        v_report_other VARCHAR2(4000);
BEGIN
        /* Call run_report_object */
        report_id:= find_report_object('reptest');
        v_report_other:= 'paramform=yes p_deptno='||:DEPT.DEPTNO;
        RUN_REPORT_OBJECT_PROC(report_id,
                                    :parameter.reportsserver,
                                                'HTMLCSS',
                                                        CACHE,

'reptest.rdf',
v_report_other,

'/reports/rwservlet');
END;
```

Example 7: Calling Oracle reports using the Run_Report_Object_Proc generic PL/SQL procedure

The PL/ SQL program unit RUN_REPORT_OBJECT_PROC also handles the secure case where Forms and Reports both run in Single Sign-On mode. In this case the database connect information is not added to the hidden runtime parameters of the Reports parameter form.

Please read the Whitepaper "Oracle Forms Services – Using Run_Report_Object() to call Reports with a parameter form" to learn about securing Reports parameter forms if not using Single Sign-On.

---

[13] If p_deptno also is shown as a list of values in the Reports parameter form, then the value passed from Forms is shown as the selected value. This is a good way of defining default selections.

```
PROCEDURE RUN_REPORT_OBJECT_PROC (report_id  REPORT_OBJECT,
                                  report_server_name    varchar2,
                                  report_format         varchar2,
                                  report_destype_name   number,
                                   report_file_name     varchar2,
                                  report_otherparam     varchar2,
                                  reports_servlet       varchar2
                                ) IS
       report_message        VARCHAR2(100)         :='';
       rep_status            VARCHAR2(100)         :='';
       vjob_id               VARCHAR2(4000)        :='';
       hidden_action         VARCHAR2(2000)        :='';
       v_report_other        VARCHAR2(4000)        :='';
       i                     number  (5);
       c                     char;
       c_old                 char;
       c_new                  char;
1      sso_user              VARCHAR2(100);


BEGIN
       -- setting Reports runtime parameters
2      SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_COMM_MODE,SYNCHRONOUS);
       SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_FILENAME,report_file_name);
       SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_SERVER,report_server_name);
       SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESTYPE,report_destype_name);
       SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESFORMAT,report_format);
3
       -- creating string for pfaction parameter

       hidden_action := hidden_action ||'&report='
       ||GET_REPORT_OBJECT_PROPERTY(report_id,REPORT_FILENAME);

       hidden_action  := hidden_action ||'&destype='
       ||GET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESTYPE);

       hidden_action  := hidden_action ||'&desformat='
       ||GET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESFORMAT);


       -- determine whether single sign-on is used or not. In the case
       -- of single sign-on add the ssoconn parameter to the otherparam section.
```

```
4       -- If run in normal mode, just use userid

        sso_user := get_application_property(sso_userid);

        IF (length(sso_user) > 0) THEN
        -- sso is used
                hidden_action := hidden_action||'&ssoconn='||:parameter.config;

        ELSE
5               -- no sso
                hidden_action  := hidden_action ||'&userid=
                                    ||get_application_property(username)||'/'
                                    ||get_application_property(password)||'@'
                                    ||get_application_property(connect_string);

        END IF;

        -- report other parameters are passed as key value pairs
        -- "key1=value1 key2=value2 ..."
        -- the following loop replaces the delimiting blank with an '&'
        -- used on the Web.
        -- c_old is initialized with a dummy value
        c_old :='@';
6
        FOR i IN 1..LENGTH(report_otherparam) LOOP

                c_new:= substr(report_otherparam,i,1);

                IF (c_new =' ') THEN
                        c:='&';
                ELSE
                        c:= c_new;
                END IF;
                -- eliminate multiple blanks
                IF (c_old =' ' and c_new = ' ') THEN
                        null;
                ELSE
                        v_report_other := v_report_other||c;
                END IF;
                -- save current value as old value
                c_old := c_new;

        END LOOP;

        hidden_action := hidden_action ||'&'||v_report_other;

        -- report_servlet contains the full path to the Reports Servlet
        -- Example1, Forms and Reports are on the same host
        -- reports_servlet:='/reports/rwservlet'
        -- Example2, Forms and Reports are on separate hosts
```

```
        -- reports_servlet:='http://host:port/reports/rwservlet'

        hidden_action := reports_servlet||
        '?_hidden_server='||report_server_name||encode(hidden_action);
7
8       -- Set pfaction to the report
        SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_OTHER,'pfaction='
        ||hidden_action||' '||report_otherparam);

        report_message:= run_report_object(report_id);
        rep_status     := report_object_status(report_message);

        IF rep_status='FINISHED' THEN
                vjob_id := substr(report_message,
                length(report_server_name)+2,length(report_message));
                WEB.SHOW_DOCUMENT(reports_servlet||'/getjobid'||vjob_id
                ||'?server='||report_server_name,' _blank');
        ELSE
                --handle errors
                message ('Error');
        END IF;

END;
```

Code review

1 – The sso_user variable helps to determine whether or not Forms runs in Single Sign-On mode.

2 – The Report Object node is populated with the Reports system command line arguments. The execution mode is synchronous, which means that the Forms client waits for the Reports Service to finish.

3 – The pfaction parameter determines the Reports command line parameters that are contained in the Reports parameter form.

4 – If the sso_user name length is greater than 0, Single Sign-On is used. If Forms runs in Single Sign-On mode, the ssoconn parameter, instead of the userid parameter, is added to the pfaction parameter.

5 – If no Single Sign-On is used, the userid parameter is added to the reports parameter form to establish the database connect for the reports execution.

6 – The string that gets added to the pfaction parameter is analyzed to filter out any character – like & or ; - that could cause trouble on the Web. Problematic characters are replaced by their hexadecimal equivalent

7 – The *encode* function is a PLSQL program unit in Forms that converts characters into their hexadecimal representation. The *encode* function is listed below.

8 – The pfaction parameter is set using the Set_Report_Object_Property(report_id, REPORT_OTHER,'…') Built-in..

Because pfaction contains the database username and passwordinformation, if not Single Sign-On is used, you might be interested in how to better secure the generated Reports parameter form. The Whitepaper "Oracle Forms Services – Using Run_Report_Object() to call Reports with a parameter form", available on http:/ / otn.oracle.com/ products/ forms/ , provides this information for you.

Encode Function

The function converts the following characters `';','/','?',':','@','+','$',','`, and `' '` into their hexadecimal equivalents if contained in the functions `URL_PARAMS_IN` argument. It can be downloaded from OTN using the following URL:

http://otn.oracle.com/products/forms/pdf/10g/frmrepparamform.zip

```
FUNCTION ENCODE (URL_PARAMS_IN Varchar2) RETURN VARCHAR2 IS


  v_url       VARCHAR2(2000) := URL_PARAMS_IN;       -- Url string
  v_url_temp  VARCHAR2(4000) :='';                   -- Temp URL string
  v_a         VARCHAR2(10);
  -- conversion variable
  v_b         VARCHAR2(10);
  -- conversion variable
  c           CHAR;
  i           NUMBER(10);

BEGIN

    FOR i IN 1..LENGTH(v_url) LOOP

      c:= substr(v_url,i,1);

      IF c in (';', '/','?',':','@','+','$',',',' ') THEN

       v_a := ltrim(to_char(trunc(ascii(substr(v_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(v_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 IF;

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

       ELSE

        v_url_temp :=v_url_temp||c;

       END IF;

   END LOOP;

       return  v_url_temp;

END;
```

Oracle Application Server 10*g* -Integrating Oracle Reports in Oracle Forms Services applications
May2004
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.