Oracle Forms Services – Using Run_Report_Object() to call Reports with a parameter form

An Oracle Technical Whitepaper
February 2004

ORACLE

Oracle Forms Services – Using Run_Report_Object() To Call Reports with a Parameter Form

Introduction....3
Problem description....3
Solving the problem with 'pfaction'....3
USING PL/ SQL functions to encode URL parameters....4
Building a generic procedure to call Reports....6
Advanced Reports Security....11
Using the oracle.reports.utility.FrmReportsInteg Bean in Forms....11
Forms Services Configuration....13
Formsweb.cfg file....13
forms90/ java directory....14
basejini.htm....14
Summary....14
Appendix A: FrmReportsInteg Bean functionality....15
SET_<nn>ENCRYPTION_KEY....15
Example for Oracle9*i* Reports....15
Example for Oracle10*g* Reports....15
SET_MAX_AGE....15
SET_COOKIE_DOMAIN....15
SET_COOKIE_PATH....16
WRITE_LOGOUTPUT....16
Enabling debug messages example....16
Disabling debug messages example....16
WRITE_USERID_COOKIE....16
Appendix C: Known Issues....17
JInitiator version dependency....17

Oracle Forms Services – Using Run_Report_Object() to call Reports with a parameter form

INTRODUCTION

To learn about using Oracle Forms' RUN_REPORT_OBJECT Built-in to call Oracle Reports from Forms, see the Whitepaper "Integrating Oracle9iAS Reports Services in Oracle9iAS Forms Services" .

This document explains how to call Reports that have a parameter form from Forms Services using the Forms RUN_REPORT_OBJECT Built-in.

The solution described in this document works with the Forms and Reports components of Oracle Application Server releases 9.0.2.x and 10*g*.

All PL/ SQL and Java code shown or mentioned in this document can be downloaded from http:/ / otn.oracle.com/ products/ forms/ .

PROBLEM DESCRIPTION

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.

The reason for Reports parameter forms not working, when called from Forms using the RUN_ REPORT_OBJECT Built-in, is an empty action attribute in the generated Reports HTML parameter form. Because RUN_REPORT_OBJECT() calls Reports directly on the server, Reports cannot access the Web environment to obtain the information required to populate the action attribute when generating the HTML parameter form.

The <ACTION> attribute is part of the standard HTML <FORM> tag that defines what to do when a user presses the submit button. The <ACTION> attribute in the Reports parameter form should contain hidden runtime parameters that are required to process the Reports request after the user presses the submit button.

This paper presents a Reports command line parameter that helps solve this problem with minimal coding.

SOLVING THE PROBLEM WITH 'PFACTION'

"pfaction" is a command line parameter in Oracle Reports that can be used to add the hidden runtime parameter to a Reports parameter form when calling Reports from Forms, using the RUN_REPORT_OBJECT() Built-in.

The syntax for the value "pfaction" parameter looks like this:

<request URL_to_rwservlet>?_hidden_<encoded_original_url_query_string>

The "request URL_to_rwservlet" contains the protocol, the host name, the port, the Reports Web context path and the Reports Servlet name.

For example:

http://fnimphiu-pc.us.oracle.com:7779/reports/rwservlet

The "encoded_original_url_query_string" is a duplicate of all Reports system and application parameters passed from Forms to Reports using the SET_REPORT_OBJECT_PROPERTY Built-in, encoded in a URL.

For example, the Reports command line parameters

destype=cache desformat=htmlcss userid=scott/tiger@orcl

would be added as a value to the "pfaction" parameter as

destype=cache%20desformat=htmlcss%20userid=scott%2Ftiger%40orcl

In order to call a Report that contains a parameter form from Oracle Forms using RUN_REPORT_OBJECT(),do the following:

1. Provide the protocol, the host name of the server, and the server port;
2. Provide the Reports virtual path and the name of the Reports Servlet;
3. URL encode the parameter value of the "pfaction" command parameter.

The following Forms Built-in will be used to pass the "pfaction" command from OracleForms Services to Reports:

SET_REPORT_OBJECT_PROPERTY(hdl,REPORT_OTHER,'pfaction …');

Note that if you are using the REPORT_OTHER Built-in to pass application parameters to Reports, the application parameters must also be contained in the pfaction parameter.

USING PL/SQL FUNCTIONS TO ENCODE URL PARAMETERS

Because the "pfaction" parameter is added as ACTION parameters to the Reports HTML parameter form, it is important to make sure that no character is contained that could cause problems. One example that may cause a problem when embedded in a URL is a blank character. Therefore, most developers URL-encode blanks as '%20'. The PL/ SQL function "ENCODE", as listed below, is an example routine that encodes the following characters: ";","/ ","?",":","@","+", "$",","," and " " (semicolon, forward slash, question mark, colon, at, plus sign, dollar sign, comma, and blank space).

```
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;
```

Figure 1: PL/SQL function to URL-encode strings

BUILDING A GENERIC PROCEDURE TO CALL REPORTS[1]

Calling Oracle Reports using RUN_REPORT_OBJECT() is best handled by a generic PL/ SQL procedure in Forms.

To successfully call a Report from Forms, the following minimum information needs to be passed to the Reports pfaction command:

- Desformat, to determine the Reports output format
- Destype, to determine the output device, (printer or cache)
- Userid, to connect Reports to the database
- Reports file name, to specify the Reports module to be executed
- Paramform = yes, to enable the Reports parameter form to be shown in the client browser

The generic PL/ SQL procedure to handle calls to Oracle Reports using the Forms RUN_REPORT_OBJECT() Built-in does require a Reports Object node to be created in Forms. One Reports Object node can be used to run any Report.

The following arguments are expected by this procedure (**1**):

report_id	The Report Object as obtained by a call to FIND_REPORT _OBJECT('<name>');
report_server_name	The name of the Reports Server, for example: "Repserv10g@fnimphiu-pc"
report_format	HTML, HTMLCSS, PDF, RTF, POSTSCRIPT, XML
report_destype_name	CACHE, FILE, MAIL, PRINTER
report_file_name	The Reports source module with or without extension
report_other_param	Any other parameter like paramform or custom application parameters. If printing to FILE or MAIL, desname needs to be provided as 'otherparam'
report_servlet	The Web access path for the Reports Servlet. The access path – if not specified as relative – must contain the protocol (http or https), the host name, the port, the Reports virtual path ('reports' for Oracle Reports 9*i* and 10*g*) and the name of the Servlet.

[1] This Whitepaper does not explain how to use the RUN_REPORT_OBJECT() Built-in in Forms. Please refer to the Whitepaper "Integrating Oracle9iAS Reports Services in Oracle9iAS Forms Services" for detailed information..
Release 9i and 10g: http:/ / otn.oracle.com/ products/ forms/ pdf/ frm9isrw9i.pdf

```
PROCEDURE RUN_REPORT_OBJECT_PROC(
1                       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;

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);

  -- creating string for pfaction parameter

  hidden_action := hidden_action ||'&report='||
3 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);

  hidden_action     := hidden_action ||'&userid='
                          ||get_application_property(username)||'/'||
                          get_application_property(password)||'@'||
                          get_application_property(connect_string);

  -- 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. This replacement does only work for values that don't
  -- include blanks themselves. If this is a requirement, then you need to
  -- customize this code accrodingly


  -- c_old is initialized with a dummy value
```

```
 c_old :='@';

FOR i IN 1..LENGTH(report_otherparam) LOOP

        c_new:= substr(report_otherparam,i,1);
4       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 or cgi (Reports6).
-- Example1, Forms and Reports are on the same host
5 -- 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);

SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_OTHER,'pfaction='||
                                        hidden_action||' '||report_otherparam);
6 -- run Reports

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
7         (report_message));

  WEB.SHOW_DOCUMENT(reports_servlet||'/getjobid'||vjob_id||'?server='||
  report_server_name,' _blank');

ELSE

8   --handle errors

    null;

END IF;

END;
```

Figure 2: Generic PL/SQL routine calling RUN_REPORT_OBJECT Built-in

The SET_REPORT_OBJECT() Built-in is used to set the runtime parameters for the Reports to run (**2**). All information provided in these calls must also be provided in the hidden "pfaction" parameter.

To define the "pfaction" parameter value, the Report object properties are read back using the GET_REPORT_OBJECT_PROPERTY() Built-in (**3**). Although you could pass the parameters in as an argument to the RUN_REPORT_ OBJECT_PROC procedure, not using them makes this solution more generic.

REPORT_OTHER parameters are key-value pairs that are delimited by a blank. The blank needs to be replaced by an ampersand '&'. This PL/ SQL routine not only replaces the blank, but also removes blanks that are not needed.

The limitation of this example is that it doesn't account for the case where values of a key-value pair contain blanks, too. However you can address this by adding custom PL/ SQL code when reusing this code. (**4**).

The value of the "pfaction" parameter must start with an ampersand '&' because the value is appended to existing parameters in the ACTION parameter of the FORM tag of the requested Report form. (**5**).

The "reports_servlet" parameter contains the Web access path to the Reports Servlet that executes the subsequent call issued by the user pressing the submit button. The ENCODE function is called to URL convert ";", "/ ","?","=","@", "+", "$",","," and " " characters (**6**)[2].

RUN_REPORT_OBJECT is called (**7**) and the Reports output is brought to the client browser using the Forms WEB.SHOW_DOCUMENT() Built-in (**8**).

Assuming that there exists a Reports Object "reptest" in the Forms module, the following code can be used in a WHEN-BUTTON-PRESSED trigger to execute a Report that has a parameter form:

[[[are you missing a code example here, or meant to point to something else?]]]

The generic PL/ SQL procedure requires the Reports Object to be passed as an argument. The Reports Object of a given Reports Object name can be obtained using the FIND_REPORT_OBJECT() Built-in (**9**).

Because it is assumed that the server running Forms Services also hosts the Reports Services, no host name needs to be passed as an argument for the Reports Servlet path variable. Instead '/ reports/ rwservlet' can be used as a relative addressing (**10**). Paramform=yes makes Reports first expose its parameter form (**11**). The parameter form always is in HTML even if the Reports destination format is PDF.

---

[2] The Oracle database also provides a URL encoding function in its UTL_URL package. The name of this function is ESCAPE.

```
9   -- Find the id of the Reports Object in Forms
    report_id:=FIND_REPORT_OBJECT('reptest');

    -- Call the generic PL/SQL procedure to run the Reports

    RUN_REPORT_OBJECT_PROC(   report_id,
10                             'repserv10g@fnimphiu-pc',
                               'HTMLCSS',
                              CACHE,
                              'REPTEST',
11                            'paramform=yes',
                              /reports/rwservlet');
```

Figure 3: Executing a Report that has a parameter form. The Reports Server runs on the same machine that hosts Forms Services

The procedure requires at least the "paramform" parameter to be passed as an argument for the OTHER_PARAM property. In the example below, an additional parameter is passed to Reports (**12**). Note that the delimiter is a blank character between the first key-value pair and the second.

```
    -- Find the id of the Reports Object in Forms
    report_id:= find_report_object('reptest');

    -- Call the generic PL/SQL procedure to run the Reports

    RUN_REPORT_OBJECT_PROC(   report_id,
                               'repserv10g@fnimphiu-pc',
                               'HTMLCSS',
                              CACHE,
                              'REPTEST',
12                            'paramform=no p_deptno=10',
                               '/reports/rwservlet');
```

Figure 4: Executing a Reports that has no parameter form but requires p_deptno to be passed as a runtime parameter

If Reports Services runs on a different machine than Forms Services, you'll need to provide the absolute access URL for the Reports Servlet (**13**).

```
    -- Find the id of the Reports Object in Forms
    report_id:= find_report_object('reptest');

    -- Call the generic PL/SQL procedure to run the Reports


    RUN_REPORT_OBJECT_PROC(   report_id,
                               'repserv10g@fnimphiu-pc',
                               'HTMLCSS',
                              CACHE,
                              'REPTEST',
                              'paramform=yes',
13                            'http://fooserver:7779/reports/rwservlet');
```

Figure 5: Executing a Report that has a parameter form. The Reports Server runs on a different server than Forms Services

Note that these examples work with the PL/ SQL procedure RUN_ REPORT_OBJECT_PROC, which is explained elsewhere in this document. You are free to create your own PL/ SQL code to handle the "pfactions" command in your applications.

ADVANCED REPORTS SECURITY[3]

Using the Reports HTML parameter form, the userid information does not appear in the Reports URL, but is contained in the hidden ACTION attribute of the Reports HTML parameter form.

Having the userid shown in the Reports HTML parameter form, although hidden, may violate the security policies of many companies. To avoid exposing the userid parameter at all, the userid connect string can be encrypted and stored in a temporary cookie on the client browser. This means the following for Reports to run:

1. The userid parameter is left empty in the Reports HTML parameter form and doesn't show in the requested URL
2. The userid connect string is encrypted and stored as a temporary cookie. The cookie is deleted immediately when closing the browser
3. The cookie expires after 30 minutes even if the browser wasn't closed
4. The default cookie domain is derived from the host running Forms Services. This secures the cookie from applications hosted by other servers accessing this information

The Reports userid cookie can be set from Forms using a Java Bean in Forms. The "oracle.reports.utility.FrmReportsInteg" Bean handles setting the userid parameter in a cookie and can be downloaded from the same OTN page that hosts this Whitepaper.

Using the oracle.reports.utility.FrmReportsInteg Bean in Forms

For the Bean to work in Forms, it needs to be added to a Forms Canvas that is visible when calling Reports.

1. In the Forms Layout Editor, add a Java Bean container to Forms, making sure that the Bean item is created in a control block.
2. To hide the Bean on the canvas, select the Bean in the Layout editor and press F4 to open the property inspector. Set the Width and Height properties to 1, the Bevel property to Plain and set the background and foreground color to the color of the canvas
3. Set oracle.reports.utility.FrmReportsInteg as a value of the Bean Item "Implementation Class" property set. Ignore any errors shown when

[3] The solution described was tested with Oracle9i Forms and Reports as well as with Oracle10g Forms and Reports.

navigating out of the Implementation class property field. This error message may show again later on, but then can be ignored too.

4. Define the Bean item name as USERID_BEAN and close the Property Palette.
5. Open the RUN_REPORT_OBJECT_PROC procedure and add the following changes:

```
PROCEDURE RUN_REPORT_OBJECT_PROC(
[…]


 -- 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);
 /*
14
 hidden_action     := hidden_action ||'&userid='
                                    ||get_application_property(username)||'/'||
                                    get_application_property(password)||'@'||
                                    get_application_property(connect_string);
 */
 hidden_action     := hidden_action ||'&userid=';
15
 -- set userid in encrypted cookie
 -- uncomment this line for production to avoid the bean writing to the Java console
 set_custom_property('control.userid_bean',1,'WRITE_LOGOUTPUT','true');
16
 set_custom_property('control.userid_bean',1,'ADD_USERID',
                                    get_application_property(username)||'/'||
17                                  get_application_property(password)||'@'||
                                    get_application_property(connect_string));


 set_custom_property('control.userid_bean',1,'WRITE_USERID_COOKIE','10g');
18
[…]
```

Figure 6: RUN_REPORT_OBJECT_PROC sets userid in an encrypted cookie on the client browser

The PL/ SQL code, that adds the userid parameter and its connect string values to the hidden_action variable, must be commented out (**14**). Oracle Forms and Oracle Reports 9.0.2 and later require that the hidden action tag contain the userid parameter, but it can be left blank or contain a dummy value. The reason for this is that Reports can be run against data sources other than the database, so that the absence of the userid parameter does not necessarily mean that a logon form

appear. In this example we add "userid=" to the hidden_action variable (**15**). The cookie is set using the Forms SET_CUSTOM_PROPERTY() Built-in.

The first call to SET_CUSTOM_PROPERTY() enables debug messages to be written to the JInitiator console, which may prove useful during design time. This should be commented out during production. (**16**).

The second call to SET_CUSTOM_PROPERTY() sets the connect string information to the Bean, which needs to precede writing the cookie to the client browser. (**17**). Finally, the cookie is written to the client browser using another call to SET_CUSTOM_PROPERTY(). The argument '10g' specifies the version of the Reports Server and determines the cookie format. For Oracle9i Reports the argument is '9i' (**18**). This sets the cookie to the client browser using the following cookie settings:

1. Expiry is set to temporary, which means that the cookie expires when the user closes the browser.
2. Validness is set to 30 minutes by default. This means that the cookie expires after 30 minutes even if the user doesn't close the browser. The Reports Server checks if the cookie is still valid before executing the Report. Because the cookie time is determined on the Forms client, it is important to consider time zone differences between the client and the server (see Appendix A).
3. The cookie path is set to '/ ' which means that all applications that run on a server in the same domain as the server running Forms Services can access this cookie (see Appendix A).
4. The cookie domain is set to the domain of the server running Forms Services. If the server domain is us.oracle.com, then only those servers that run in this domain can access the client side cookie (see Appendix A).
5. The default Reports key is used to encrypt the information. This default value is "reports9i" for Oracle9*i* and Oracle10*g* Reports (see Appendix A).

Forms Services Configuration

To deploy the FrmReportsInteg Bean with Forms, changes are required in the formsweb.cfg file and the basejini.htm file, both located in the <Oracle Home>\ forms90\ server directory.

Formsweb.cfg file

The archive file fmrRwInteg.jar that contains the FrmReportsInteg Bean needs to be configured for download when the Forms application is started. Add the following line to the named configuration section for your application in the formsweb.cfg file, located in the forms90/ server directory:

```
[ <name> ]
…
archive_jini= f90all_jinit.jar,frmrwinteg.jar
...
```

forms90/java directory

Make sure that the frmrwinteg.jar file is located in the forms90/ java directory of your Forms Services installation.

You can tell from looking at the JInitiator Java console on the client if this configuration works. If you see a Java error printed to the console that reports a missing Java class file, then this is most likely a misconfiguration.

basejini.htm

For the Forms Applet to get permissions to set the temporary authentication cookie, the MAYSCRIPT parameter needs to be set in the basejini.htm template file located in the forms60/ sever directory.

Internet Explorer section of basejini.htm:

<PARAM NAME="MAYSCRIPT" VALUE="true">

Netscape section of basejini.htm

MAYSCRIPT=true

You can tell from looking at the JInitiator Java console on the client if this configuration works. If you see a Java error printed to the console that reports a missing Java class file, then this is most likely a misconfiguration.

SUMMARY

This document shows a simplified and secure solution for Reports that are called from Forms using RUN_REPORT_OBJECT(), but require a Reports parameter form to be exposed first. The Reports command line parameter 'pfaction' is used to create a working Reports HTML parameter form, while a Java Bean in a Form can be used to provide additional security for this solution.

Oracle Forms and Oracle Reports development teams are currently working on providing native functionality described in this paper. Until then the solution described in this document provides the same service.

APPENDIX A: FRMREPORTSINTEG BEAN FUNCTIONALITY

The Java Bean used to set the cookie provides convenience methods for users that don't want to use the default values but need to modify some of the cookie settings.

SET_<nn>ENCRYPTION_KEY

The SET_<nn>ENCRYPTION_KEY property allows the application developer to issue another key for encrypting the Reports cookie other than the default. Before changing the key in the cookie, make sure that the key is also changed in the Reports Server rwservlet.properties file (Reports9*i* and Reports 10*g*).

Example for Oracle9*i* Reports

```
set_custom_property('control.userid_bean',1,'SET_9iENCRYPTION_KEY',
'myOwnKeyFor9i');
```

Example for Oracle10*g* Reports

```
set_custom_property('control.userid_bean',1,'SET_10gENCRYPTION_KEY',
'myOwnKeyFor10g');
```

The key, once changed, is kept until the end of the Java Bean session. If you are okay with the default encryption key, don't use this property.

SET_MAX_AGE

The SET_MAX_AGE property allows the application developer to specify a time in minutes for which the Reports userid cookie is valid (unless the user closes the browser). The cookie expiration is determined on the Reports Server. The default value is 30 minutes.

The SET_MAX_AGE property can be used to handle time zone differences between the server and the client. For example, if the Reports Services is installed in the USA and the client runs in Europe, then, because the time stored in the cookie is determined on the client, to set the cookie validness to 30 minute you need to consider an offset of 9 hours

```
set_custom_property('control.userid_bean',1,SET_MAX_AGE,570);
```

To set the MAX_AGE_PROPERTY to 6o minutes for a Reports Server and Forms client that run in the same timezone use

```
set_custom_property('control.userid_bean',1,SET_MAX_AGE,60);
```

SET_COOKIE_DOMAIN

The cookie domain defines the scope of servers, from where hosted applications can access the cookie information if requested by the user. The minimum requirement is a domain that has at least has two '.' in it. For example, '.oracle.com' is a valid domain while 'oracle.com' isn't. The domain can be set to a

complete server name, therefore ensuring that only applications started on this server can access the cookie.

set_custom_property(’control.userid_bean’,1,SET_COOKIE_DOMAIN, ’.oracle.com’);

The default value for cookie domain is the domain of the server that runs the Forms Servlet.

To reset the default domain, use

set_custom_property('control.userid_bean',1,SET_COOKIE_DOMAIN, ’**reset**’);

SET_COOKIE_PATH

The cookie path defines the virtual path an application needs to access to the client side cookie. By default the path value is set to ‘/ ’, which means that applications downloaded from any virtual path in the cookie’s domain can access the cookie. To restrict access to only those applications downloaded from a specific virtual path, like “reports”, use the following Java Bean functionality:

set_custom_property('control.userid_bean',1,SET_COOKIE_PATH, ’/ reports/ ’);

The combination of cookie_domain and cookie_path are used to determine access to the cookie. Setting the domain to ‘.us.oracle.com’ and the cookie_path to ‘/ reports/ ’ allow only applications accessible via http:/ / <server name> .us.oracle.om:<port> / reports/ to access the cookie information.

WRITE_LOGOUTPUT

The WRITE_LOGOUTPUT property allows to switch on and off debug messages written to the client side JInitiator console. By default no debug message is written.

Enabling debug messages example

set_custom_property('control.userid_bean',1, WRITE_LOGOUTPUT,’true’);

Disabling debug messages example

set_custom_property('control.userid_bean',1, WRITE_LOGOUTPUT,’false’);

WRITE_USERID_COOKIE

The 'WRITE_USERID_COOKIE property actually sets the cookie to the client browser. This property needs to be called whenever the user database connect information for a Reports changes or after the cookie has been invalidated by exceeding the time defined as max_age.

To write a cookie for Oracle9i Reports

```
set_custom_property('control.userid_bean',1,'WRITE_USERID_COOKIE','9i');
```

To write a cookie for Oracle10g Reports

```
set_custom_property('control.userid_bean',1,'WRITE_USERID_COOKIE','10g');
```

APPENDIX C: KNOWN ISSUES

JInitiator version dependency

For this solution to work, JInitiator of version 1.3.1.13 or above should be used. Problems have been reported when using earlier JInitiator version.

Oracle Forms Services – Using Run_Report_Object() to call Reports with a parameter form
February 2004
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.