Oracle Forms Services – Secure Web.Show_Document() calls to Oracle Reports

An Oracle Technical Whitepaper
February 2004

Secure Web.Show_Document() calls to Oracle Reports

Secure Web.Show_Document() calls to Oracle Reports

INTRODUCTION

Using the Oracle Forms Web.Show_Document() Built-in to call Oracle Reports on the Web is an alternative to the Run_Report_Object() Built-in. The Web.Show_Document() Built-in accesses Web resources by issuing a HTTP "GET" request from the browser URL. HTTP "GET" requests, in contrast to "POST" requests, show the complete URL string with all the request parameters in the browser's address bar, including those parameters that are considered sensitive information, such as logon information.

This Whitepaper shows you how to secure calls to Oracle Reports Services, issued by Forms using the Web.Show_Document() Built-in, by eliminating the need to expose the sensitive userid information in the Reports request URL.

The solution described in this document is based on a Java Bean that resides on the Oracle Forms Web client and works with the Forms and Reports components of Oracle Application Server releases 9.0.2.x and 10*g*. The Java Bean and its source code can be downloaded with this paper.

USING WEB.SHOW_DOCUMENT BUILT-IN TO CALL REPORTS

This section briefly covers the use of the Forms Web.Show_Document() Built-in to call Oracle Reports on the Web.

Web.Show_Document syntax

The Forms Web.Show_Document() Built-In requires two arguments passed within the call

```
Web.Show_Document(URL, Target);
```

URL – The URL is passed as a string in a variable, or as a combination of both. If the target Web page is located on the same server that runs Forms Services, relative addressing could be used.

Target – Definition of the target where the addressed Web page should be displayed. Values must be single-quoted. Possible target values are '_blank' to show the Reports output in an extra browser window, '_self' to replace the Forms application with the Reports output, '<frame name>' to load the Reports output into a named frame of the multi frame HTML page.

Calling Oracle Reports on the Web

After installing Oracle Application Server, Oracle Reports Services can be accessed by the following URL

http(s)://<server>:<port>/reports/rwservlet?<reports query parameters>

A complete syntax example to run Reports from a browser looks like this

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

server – the name of the Reports Server[1] used. This name can be omitted to use the in process Reports Server that is automatically started by the Reports Servlet.

report – the name of the Reports module to execute

desformat – the output format of the returned Reports result set. Desformat can be htmlcss, html, pdf, xml, rtf and delimited. For Reports run from Forms pdf and htmlcss are the most commonly used options

destype – determines where the Reports output gets written to. "Cache" specifies that the Reports output gets streamed to the requesting browser. '

userid – in the case of a Reports that needs to query a database for its data, the userid parameter contains the username, the user password and the connection information for the database.

paramform – determines if Reports should display a HTML parameter form before executing the request. The parameter form can be used for the user to further filter the expected Reports result set. Valid values are 'yes' and 'no'.

To reduce the length of the Reports request URL, you can create a key entry in the Reports cgicmd.dat configuration file to store command line parameters that don't change from one Report to the other. In this case the first argument in a Reports Web request, right after the question mark, must be the key name[2].

Calling Reports from Forms using Web.Show_Document

The following PL/ SQL example assumes the Reports Services to run on the same server that hosts the Forms Services, thus using relative addressing. The Reports output is formatted in HTML (desformat=htmlcss) and no Reports parameter form is shown before running the report (paramform=no). To filter the Reports result set, a user parameter is passed to Reports, specifying the department id to retrieve information for (p_deptno=10).

---

[1] Please refer to the Reports Services documentation on how to create a Reports Service
[2] You can also specify the userid parameter in the cgicmd.dat file and thus hide it from the URL. However the userid will show in the HTML source code of the parameter form if used

When calling Web.Show_Document(), the second argument is specified as '_blank', which means that the Reports output is shown in a separate browser window.

```
DECLARE

  rep_url  varchar2(2000);

BEGIN

  rep_url:='/reports/rwservlet?server=repserv10g@fnimphiu-pc&report=reptest.rdf'
           ||'&desformat=htmlcss&destype=cache&userid=scott/tiger@orcl'
           ||'&p_deptno=10&paramform=no';

  WEB.SHOW_DOCUMENT(rep_url,'_blank');

END;
```

Example 1: PL/SQL Example using Web.Show_Document() Built-in to call Oracle Reports

The sensitive information for the "userid" parameter is added to the Reports request URL and will be shown in the browser.

SECURE WEB.SHOW_DOCUMENT[3] CALLS TO ORACLE REPORTS

One option to secure Reports called by the Forms Web.Show_Document() Built-in is to use run Forms and Reports in single sign-on mode. The Oracle Application Server provides an integrated single sign-on solution that can be used without any changes required to the Forms and Reports application modules[4].

Another option is the technique explained in the rest of this paper.

Adding the userid parameter to the Reports request URL violates the security policies of many companies. Thus, to avoid exposing the userid parameter at all, the userid connect string must 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 solution described was tested with Oracle9i Forms and Reports as well as with Oracle10g Forms and Reports.
[4] Using single sign-on with Forms and Reports is explained in the Oracle9i Forms and Reports integration Whitepaper, available at otn.oracle.com/ products/ forms

3. The cookie expires after 30 minutes even if the browser isn’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. A Bean that performs this action, “oracle.reports.utility.FrmReportsInteg”, has been written to accompany this Whitepaper, and handles setting the userid parameter in a cookie. The Bean is contained in a jar file called “frmrwinteg.jar” and can be downloaded with this document from http:/ / otn.oracle.com/ products/ forms.

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 the value “oracle.reports.utility.FrmReportsInteg” for the Bean Item “Implementation Class” property . Ignore any errors shown when navigating out of the Implementation class property field. This error message may show again later on, but then can be ignored too.[5]
4. Define the Bean item name as USERID_BEAN and close the Property Palette.
5. To use the PL/ SQL code shown in Example 1, the following changes need to be done in the code to exclude the userid value from the Reports request URL. Instead the userid value is stored in a temporary cookie on the client.

[5] To avoid the error message to be shown, add the frmrwinteg.jar file name with the complete path information to the FORMS90_BUILDER_CLASSPATH registry variable.

```
DECLARE

  rep_url  varchar2(2000);

BEGIN

  rep_url:='/reports/rwservlet?server=repserv10g@fnimphiu-pc&report=reptest.rdf'
           ||'&desformat=htmlcss&destype=cache&userid='


           ||'&p_deptno='|| :dept.deptno&paramform=no';


   -- Write log messages to the Forms JInitiator console. The next line must
   -- be disabled before running this code in any production environment
   set_custom_property('control.userid_bean',1,'WRITE_LOGOUTPUT','true');

   -- set userid in encrypted cookie before calling Web.Show_Document()
   set_custom_property('control.userid_bean',1,'ADD_USERID',
                              get_application_property(username)||'/'||
                              get_application_property(password)||'@'||
                              get_application_property(connect_string));
     -- writing the cookie
     set_custom_property('control.userid_bean',1,'WRITE_USERID_COOKIE','10g');

    WEB.SHOW_DOCUMENT(rep_url,'_blank');

END;
```

Example 2: PL/SQL Example securing the Web.Show_Document() Built-in call to Oracle Reports. The userid parameter value is temporarily stored in an encrypted cookie on the client

Oracle Reports 9.0.2 and later still require that the userid parameter is added to the request URL, but the parameter value can be left blank as shown in Example 2 (userid=). All that 'userid=' parameter does is to indicate to the Reports Server that the requested report requires a database connect and that the database credentials are stored in a temporary cookie on the client.

The first call to SET_CUSTOM_PROPERTY() (Example 2) enables debug messages to be written to the JInitiator console, which may prove useful during design time. This should be disabled before productizing the application.

The second call to SET_CUSTOM_PROPERTY() sends the connect string information to the Bean, which it needs to create the cookie

Finally, the cookie is created for 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'.

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 file

For the Java Bean to work, it is required to grant permission to the Forms Applet to use scripting. Edit the basejini.htm file, or any other template file you use to launch Forms, and add the following lines to the IE section and Netscape section.

Internet Explorer

```
<OBJECT …>

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

</OBJECT>
```

Netscape

```
<EMBED …

MAYSCRIPT=TRUE>

</EMBED>
```

SUMMARY

Calling Oracle reports from Oracle Forms using the Web.Show_Document() Built-in isn't secure as it exposes sensitive information in the browser URL. This document helps to secure sensitive information by using an encrypted temporary cookie that can be read by the Reports Server but never is exposed to the browser URL. This solution should be seen as a solution for customers that cannot use the Run_Report_Object() Built-in for calling Oracle Reports from Forms and that don't want to leverage the Oracle Single Sign-On solution in Oracle Application Server.

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 B: EXTENDED PL/SQL EXAMPLE

The following PL/ SQL example uses all Java Bean properties provided by the FrmReportsInteg Java Bean:

- It enables log messages to be written to the JInitiator console
- It sets cookie expiry (max_age) to 60 minutes
- It sets the cookie domain to .fooserver.com
- It sets the cookie path to / reports/

```
DECLARE
  rep_url  varchar2(2000);
BEGIN

  rep_url:=‘/reports/rwservlet?server=repserv10g@fnimphiu-pc&report=reptest.rdf’
          ||’&desformat=htmlcss&destype=cache&userid=’
          ||’&p_deptno=’|| :dept.deptno&paramform=no’;


   -- Write log messages to the Forms JInitiator console.
   set_custom_property('control.userid_bean',1,'WRITE_LOGOUTPUT','true');


   -- set userid in encrypted cookie before calling Web.Show_Document()
   set_custom_property('control.userid_bean',1,'ADD_USERID',
                                   get_application_property(username)||'/'||
                                   get_application_property(password)||'@'||
                                   get_application_property(connect_string));
    -- set max age to 60 minutes. Make sure you consider time offsets
    set_custom_property(' control.userid_bean ',1,'SET_MAX_AGE',’60’);

     -- set the cookie domain to .fooserver.com
    set_custom_property('control.userid_bean’,1,'SET_COOKIE_DOMAIN',
                                ’.fooserver.com’);

     -- set the cookie path to /reports/
    set_custom_property('control.userid_bean’,1,'SET_COOKIE_PATH','/reports/');
     -- writing the cookie
     set_custom_property('control.userid_bean',1,'WRITE_USERID_COOKIE','10g');

    WEB.SHOW_DOCUMENT(rep_url,’_blank’);

END;
```

Example 3: Extended PL/SQL Example

APPENDIX C: KNOWN ISSUES

Problems have been reported for Jinitiator versions prior to JInitiator 1.3.1.13. If you experience problems, please make sure that your Jinitiator version is at least of version 1.3.1.13.

Problems can be reported on the Forms forum on http:/ / otn.oracle.com.

Secure Web.Show_Document() calls to Oracle Reports
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