Oracle Forms Services – Secure Web.Show_Document() calls to Oracle Reports Server 6i

An Oracle Technical Whitepaper
March 2004

ORACLE

Secure Web.Show_Document() calls to Oracle Reports Server 6i

Secure Web.Show_Document() calls to Oracle Reports Server 6i

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 6i and Reports 6i components of Oracle9***i*** Application Server Release 1.

SOLUTION FOR ORACLE REPORTS SERVER 6I

Oracle9i Reports Services in Oracle9***i*** Application Server 9.0.2 and above handle authentication cookies differently than Reports Server 6i. The differences are within the format of the cookie, the number of cookies set and the way cookie expiration is defined.

Securing Forms Web.Show_Document() calls to Oracle9***i*** Reports, using Oracle9***i*** Application Server Reports Services and Oracle Application Server 10g Reports Services, is the subject of separate Whitepaper available on http:/ / otn.oracle.com/ products/ forms.

Please make sure you download and implement the Oracle9i Reports version of this document when upgrading Forms 6i applications to Oracle9i Forms and beyond, or when accessing Oracle9***i***AS Reports Services with Forms 6i.

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, the Oracle Reports Server can be accessed by the following URL:

On Unix

http(s)://<server>:<port>/dev60cgi/rwcgi60?<reports query parameters>

On Windows

http(s)://<server>:<port>/dev60cgi/rwcgi60**.exe**?<reports query parameters>

The default Reports Server installation in Oracle9***i*** Application Server Release 1 (9.0.1.x) uses the Reports Common Gateway Interface (cgi) – rwcgi60 – to access Reports on the Web.

Starting with Oracle9*i*AS Report Services (Oracle9***i*** Reports), the cgi interface is deprecated and the Reports Servlet – *rwservlet* - is used instead.

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

http://<server>:<port>/dev60cgi/rwcgi60?**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

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. '

[1] Please refer to the Reports Services documentation on how to create a Reports Service

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 server OS used in this example is Unix, as you can tell from not appending '.exe' to the "rwcgi60" executable name.

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).

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:='/dev60cgi/rwcgi60?server=repserv6i&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

[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

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 CALLS TO ORACLE REPORTS

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 omitted 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 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.FrmReportsInteg608”, 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 “frmrwinteg608.jar” and can be downloaded with this document from http:/ / otn.oracle.com/ products/ forms.

Using the oracle.reports.utility.FrmReportsInteg608 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.FrmReportsInteg608” 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.[3]

[3] To avoid the error message to be shown, add the frmrwinteg608.jar file name with the complete path information to the FORMS90_BUILDER_CLASSPATH registry variable.

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.

```
DECLARE

  rep_url  varchar2(2000);

BEGIN

  rep_url:='/dev60cgi/rwcgi?server=repserv6i&report=reptest.rdf'
          ||'&desformat=htmlcss&destype=cache'


          ||'&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','');

   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

Oracle9i Reports upgrade note: Oracle Reports 9.0.2 and later releases still require that the userid parameter is added to the request URL, but the parameter value is left blank. In this case the 'userid= ' parameter indicates 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().

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. 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).
3. 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).
4. The default Reports key is used to encrypt the information.

Forms Services configuration

To deploy the FrmReportsInteg608 Bean with Forms, changes are required in the formsweb.cfg file and the basejini.htm file, both located in the <Oracle Home> \ forms60\ server directory.

Formsweb.cfg file

The archive file fmrRwInteg608.jar that contains the FrmReportsInteg608 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= f60all_jinit.jar,frmrwinteg608.jar
...
```

forms60/java directory

Make sure that the frmrwinteg608.jar file is located in the forms60/ java directory of your Forms Services installation.

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: FRMREPORTSINTEG608 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 environment variable.

Example

```
set_custom_property('control.userid_bean',1,'SET_ENCRYPTION_KEY',
'myOwnKeyFor6i');
```

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.

ADJUST_TIMEZONE_DIFFERENCE

By default, the cookie will hold the connection string for a default time period of 30 minutes, or until the browser is exited. This time is defined on the Reports Server and can be overridden by setting the REPORTS60_COOKIE_EXPIRE environment variable. It can be set in minutes for an alternative time period to either allow the cookie to last longer or expire more quickly.

Because the cookie is set on the client, the cookie creation time is the current client time. For example, the time difference of a Reports Server installed in California to a Forms Applet that runs on a Browser client in New York is 3 hours. This may exceed the time period defined on the Reports Server for the cookie validness. Therefore, timezone differences must be reflected when setting the Reports cookie on the Forms client.

The ADJUST_TIMEZONE_ DIFFERENCE property of the Forms Java bean introduced in this Whitepaper can be used to handle time differences between the Forms client and the Reports Server.

For the above example, the cookie needs to be created with a creation time three hours earlier than what is the current time in New York:

```
set_custom_property('control.userid_bean',1,ADJUST_TIMEZONE_
DIFFERENCE,'-180');
```

If The Forms Java client and the Reports Server are located in the same timezone, or if the REPORTS60_COOKIE_EXPIRE value is set big enough, you don't need to set the ADJUST_TIMEZONE_ DIFFERENCE parameter.

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 "dev60cgi", use the following Java Bean functionality:

set_custom_property('control.userid_bean',1,SET_COOKIE_PATH, '/ dev60cgi/ ');

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

set_custom_property('control.userid_bean',1,'WRITE_USERID_COOKIE','');

APPENDIX B: EXTENDED PL/SQL EXAMPLE

The following is a PL/ SQL example for the FrmReportsInteg608 Java Bean using all its properties:

- It enables log messages to be written to the JInitiator console
- It handles a timezone difference of 1 hour for a Reports Server installed in Germany and the Forms Java client running on a PC in London
- It sets the cookie domain to .fooserver.com
- It sets the cookie path to / dev60cgi/

```
DECLARE
  rep_url  varchar2(2000);
BEGIN

  rep_url:='/dev60cgi/rwcgi60?server=repserv6i &report=reptest.rdf'
          ||'&desformat=htmlcss&destype=cache'
          ||'&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));

    -- handle time zone difference
    set_custom_property(' control.userid_bean ',1,'ADJUST_TIMEZONE_
                                DIFFERENCE','-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 /dev60cgi/
    set_custom_property('control.userid_bean',1,'SET_COOKIE_PATH','/dev60cgi/');
     -- writing the cookie
     set_custom_property('control.userid_bean',1,'WRITE_USERID_COOKIE','');

    WEB.SHOW_DOCUMENT(rep_url,'_blank');

END;
```

Example 3: Extended PL/SQL Example

APPENDIX C: KNOWN ISSUES

JInitiator version dependency

For this solution to work, JInitiator of version 1.3.1.9, 1.3.1.13 and above should be used. Problems have been reported when using earlier JInitiator versions.

Secure Web.Show_Document() calls to Oracle Reports Server 6i
February 2004
Author: Frank Nimphius
Contributing Authors: A big “thank you” to Ajay Gopalan for testing this Java Bean

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