UTL\_HTTP and SSL (HTTPS) using Oracle Wallets

Since Oracle 9i Release 2, the UTL\_HTTP package has had the ability to access resources over HTTPS as well as HTTP. This article describes the method for enabling HTTPS access from the UTL\_HTTP package.

* [Access Control List (ACL)](https://oracle-base.com/articles/misc/utl_http-and-ssl#acl)
* [Test Unsecured Connection](https://oracle-base.com/articles/misc/utl_http-and-ssl#test-unsecured-connection)
* [Get Site Certificates](https://oracle-base.com/articles/misc/utl_http-and-ssl#get-site-certificates)
* [Create an Oracle Wallet Containing the Certificates](https://oracle-base.com/articles/misc/utl_http-and-ssl#create-a-wallet)
* [Test Secured Connection](https://oracle-base.com/articles/misc/utl_http-and-ssl#test-secured-connection)
* [Authentication](https://oracle-base.com/articles/misc/utl_http-and-ssl#authentication)
* [SSLv3, TLSv1 and POODLE](https://oracle-base.com/articles/misc/utl_http-and-ssl#sslv3-tlsv1-poodle)

Access Control List (ACL)

If you are using Oracle 11g, you will need to provide an ACL to allow the UTL\_HTTP package to interact with an external host. This is described here.

* [Fine-Grained Access to Network Services in Oracle Database 11g Release 1](https://oracle-base.com/articles/11g/fine-grained-access-to-network-services-11gr1)

Test Unsecured Connection

Before we start trying to configure SSL, lets see what happens if we attempt to access a HTTPS resource using the UTL\_HTTP package. To do this, create the following procedure.

CREATE OR REPLACE PROCEDURE show\_html\_from\_url (

p\_url IN VARCHAR2,

p\_username IN VARCHAR2 DEFAULT NULL,

p\_password IN VARCHAR2 DEFAULT NULL

) AS

l\_http\_request UTL\_HTTP.req;

l\_http\_response UTL\_HTTP.resp;

l\_text VARCHAR2(32767);

BEGIN

-- Make a HTTP request and get the response.

l\_http\_request := UTL\_HTTP.begin\_request(p\_url);

-- Use basic authentication if required.

IF p\_username IS NOT NULL and p\_password IS NOT NULL THEN

UTL\_HTTP.set\_authentication(l\_http\_request, p\_username, p\_password);

END IF;

l\_http\_response := UTL\_HTTP.get\_response(l\_http\_request);

-- Loop through the response.

BEGIN

LOOP

UTL\_HTTP.read\_text(l\_http\_response, l\_text, 32766);

DBMS\_OUTPUT.put\_line (l\_text);

END LOOP;

EXCEPTION

WHEN UTL\_HTTP.end\_of\_body THEN

UTL\_HTTP.end\_response(l\_http\_response);

END;

EXCEPTION

WHEN OTHERS THEN

UTL\_HTTP.end\_response(l\_http\_response);

RAISE;

END show\_html\_from\_url;

/

This procedure works for a regular HTTP resource, but what happens if we call it using a HTTPS resource? The following example uses "https://gb.redhat.com/".

SET SERVEROUTPUT ON

EXEC show\_html\_from\_url('https://gb.redhat.com/');

\*

ERROR at line 1:

ORA-29273: HTTP request failed

ORA-06512: at "SYS.UTL\_HTTP", line 1527

ORA-29261: bad argument

ORA-06512: at "TEST.SHOW\_HTML\_FROM\_URL", line 22

ORA-29273: HTTP request failed

ORA-06512: at "SYS.UTL\_HTTP", line 1130

ORA-29024: Certificate validation failure

ORA-06512: at line 1

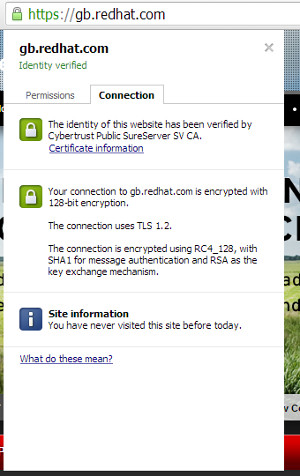
SQL>

The error stack shows the "ORA-29024: Certificate validation failure" error.

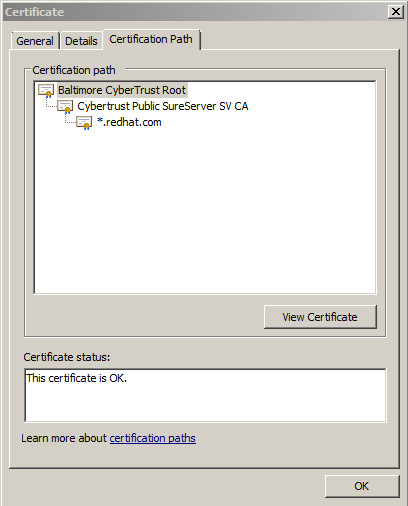
Get Site Certificates

In order to make connections to a secured resource, we need to get the necessary certificate. The easiest way to do this is using a browser. The example below uses the Chrome browser.

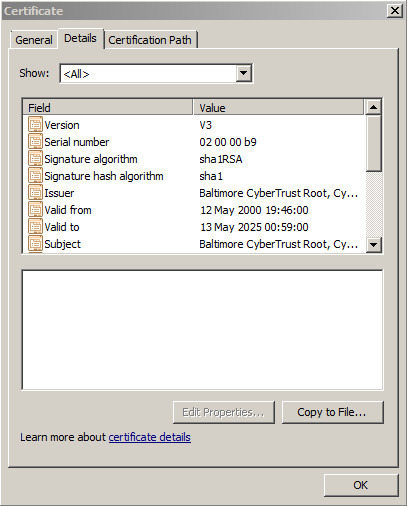
Using the browser, go to the URL you are attempting to access from PL/SQL. In this case "https://gb.redhat.com/". Click the lock icon in the URL bar to display the certificate menu and click on the "Connection" tab.



Click the "Certificate information" link and click the "Certification Path" tab on the resulting dialog.



For the root node in the "Certification path", highlight the node and click the "View Certificate" button. On the resulting dialog, click the "Details" tab and click the "**Copy to File..." button to save the certificate information**.



On the resulting wizard, do the following.

* Click the "Next" button on the welcome screen.
* Select the "**Base-64 encoded X.509 (.CER)**" option and click the "Next" button. Other formats work, but I've found this to be the most consistent.
* Enter suitable file name and click the "Next" button.
* Click the "Finish" button.

A similar dialog is displayed in Firefox by clicking "URL Icon > More Information > View Certificate > Details Tab".

 Thanks to Erik for pointing out I don't need to download the intermediate certificates. Just the root certificate.

Create an Oracle Wallet Containing the Certificates

**Create a new location to hold the wallet**.

$ mkdir -p /u01/app/oracle/admin/DB11G/wallet

Create a new wallet.

$ orapki wallet create -wallet /u01/app/oracle/admin/DB11G/wallet -pwd WalletPasswd123 -auto\_login

If the wallet password is too weak, you will get a message telling you so.

Invalid password....

PASSWORD\_POLICY : Passwords must have a minimum length of eight

characters and contain alphabetic characters combined with numbers or

special characters.

In Oracle 11.2 the same issue causes a failure to create the wallet with the following message.

Unable to save wallet at /u01/app/oracle/admin/DB11G/wallet

With the wallet created, we can add the certificate we saved earlier.

$ orapki wallet add -wallet /u01/app/oracle/admin/DB11G/wallet -trusted\_cert -cert "/host/BaltimoreCyberTrustRoot.crt" -pwd WalletPasswd123

The root certificate may fail to load with the following message, which can be ignored. It just means it was already present by default.

Could not install trusted cert at/host/Builtin Object Token:GTE CyberTrust Global Root

PKI-04003: The trusted certificate is already present in the wallet.

Test Secured Connection

We are now ready to access the secured resource, but we must provide the UTL\_HTTP package with the wallet details so it can make the secured connections. This is done using the UTL\_HTTP.SET\_WALLET procedure. Repeating the previous test now works successfully.

SET SERVEROUTPUT ON

EXEC UTL\_HTTP.set\_wallet('file:/u01/app/oracle/admin/DB11G/wallet', 'WalletPasswd123');

EXEC show\_html\_from\_url('https://gb.redhat.com/');

... HTML output removed ...

PL/SQL procedure successfully completed.

SQL>

From Oracle 11gR2 onward, if you are using the -auto\_login option on the wallet, you don't have to specify the wallet password. You just pass NULL instead of the password. Thanks to Jason in the comments for pointing this change out!

SET SERVEROUTPUT ON

EXEC UTL\_HTTP.set\_wallet('file:/u01/app/oracle/admin/DB11G/wallet', NULL);

EXEC show\_html\_from\_url('https://gb.redhat.com/');

... HTML output removed ...

PL/SQL procedure successfully completed.

SQL>

You may want to incorporate this into the procedure.

CREATE OR REPLACE PROCEDURE show\_html\_from\_url (

p\_url IN VARCHAR2,

p\_username IN VARCHAR2 DEFAULT NULL,

p\_password IN VARCHAR2 DEFAULT NULL,

p\_wallet\_path IN VARCHAR2 DEFAULT NULL,

p\_wallet\_password IN VARCHAR2 DEFAULT NULL

) AS

l\_http\_request UTL\_HTTP.req;

l\_http\_response UTL\_HTTP.resp;

l\_text VARCHAR2(32767);

BEGIN

-- If using HTTPS, open a wallet containing the trusted root certificate.

IF p\_wallet\_path IS NOT NULL AND p\_wallet\_password IS NOT NULL THEN

UTL\_HTTP.set\_wallet('file:' || p\_wallet\_path, p\_wallet\_password);

END IF;

-- Make a HTTP request and get the response.

l\_http\_request := UTL\_HTTP.begin\_request(p\_url);

-- Use basic authentication if required.

IF p\_username IS NOT NULL and p\_password IS NOT NULL THEN

UTL\_HTTP.set\_authentication(l\_http\_request, p\_username, p\_password);

END IF;

l\_http\_response := UTL\_HTTP.get\_response(l\_http\_request);

-- Loop through the response.

BEGIN

LOOP

UTL\_HTTP.read\_text(l\_http\_response, l\_text, 32766);

DBMS\_OUTPUT.put\_line (l\_text);

END LOOP;

EXCEPTION

WHEN UTL\_HTTP.end\_of\_body THEN

UTL\_HTTP.end\_response(l\_http\_response);

END;

EXCEPTION

WHEN OTHERS THEN

UTL\_HTTP.end\_response(l\_http\_response);

RAISE;

END show\_html\_from\_url;

/

Authentication

If you are accessing a site that requires authentication, you will need to do one of two things depending on the type of authentication used.

If the site uses basic authentication, simply specify the credentials in the call to SHOW\_HTOM\_FROM\_URL, which will use them in the UTL\_HTTP.SET\_AUTHENTICATION call.

SET SERVEROUTPUT ON

EXEC UTL\_HTTP.set\_wallet('file:/u01/app/oracle/admin/DB11G/wallet', 'WalletPasswd123');

EXEC show\_html\_from\_url('https://gb.redhat.com/', 'username', 'password');

... HTML output removed ...

PL/SQL procedure successfully completed.

SQL>

If the page uses digest authentication, then you will need to will need to install the [digest\_auth\_api](https://oracle-base.com/dba/script?category=miscellaneous&file=digest_auth_api.sql) package, then make the following modification to the test code.

CREATE OR REPLACE PROCEDURE show\_html\_from\_url (

p\_url IN VARCHAR2,

p\_username IN VARCHAR2 DEFAULT NULL,

p\_password IN VARCHAR2 DEFAULT NULL,

p\_wallet\_path IN VARCHAR2 DEFAULT NULL,

p\_wallet\_password IN VARCHAR2 DEFAULT NULL

) AS

l\_http\_request UTL\_HTTP.req;

l\_http\_response UTL\_HTTP.resp;

l\_text VARCHAR2(32767);

BEGIN

-- If using HTTPS, open a wallet containing the trusted root certificate.

IF p\_wallet\_path IS NOT NULL AND p\_wallet\_password IS NOT NULL THEN

UTL\_HTTP.set\_wallet('file:' || p\_wallet\_path, p\_wallet\_password);

END IF;

-- Make a HTTP request and get the response.

l\_http\_request := digest\_auth\_api.begin\_request(p\_url => p\_url,

p\_username => p\_username,

p\_password => p\_password,

p\_method => 'GET');

l\_http\_response := UTL\_HTTP.get\_response(l\_http\_request);

-- Loop through the response.

BEGIN

LOOP

UTL\_HTTP.read\_text(l\_http\_response, l\_text, 32766);

DBMS\_OUTPUT.put\_line (l\_text);

END LOOP;

EXCEPTION

WHEN UTL\_HTTP.end\_of\_body THEN

UTL\_HTTP.end\_response(l\_http\_response);

END;

EXCEPTION

WHEN OTHERS THEN

UTL\_HTTP.end\_response(l\_http\_response);

RAISE;

END show\_html\_from\_url;

/

You can then call the test code in the same way you did for basic authentication.

SET SERVEROUTPUT ON

EXEC show\_html\_from\_url('https://gb.redhat.com/', 'username', 'password', '/u01/app/oracle/admin/DB11G/wallet', 'WalletPasswd123');

... HTML output removed ...

PL/SQL procedure successfully completed.

SQL>

SSLv3, TLSv1 and POODLE

With the publicity about the [POODLE bug](http://www.symantec.com/connect/blogs/ssl-30-vulnerability-poodle-bug-aka-poodlebleed), many web masters are turning off SSLv3 support. Depending on your Oracle database version/patch, that can present a bit of a problem for people using UTL\_HTTP to access HTTPS resources, as described here.

* UTL\_HTTP Package Fails With ORA-29273 ORA-28860 When Using TLSv1 ([Doc ID 727118.1](https://support.oracle.com/epmos/faces/DocContentDisplay?id=727118.1)) : Basically, older database releases only allow HTTPS using the SSLv3 protocol from UTL\_HTTP. If you want to use the TLSv1 protocol you need to make sure you are on a patched up version of 11.2.

 The MOS note for the following comment has been removed/hidden, so it's possible this was a bug that is now fixed in 12.1.0.2.

Interestingly, if you upgrade to Oracle 12c, you might have problems in the other direction, since Oracle 12c prevents UTL\_HTTP calls over HTTPS to anything older than TLSv1.2, as described here.

* UTL\_HTTP Gives Error Over HTTPS Using RDBMS 12.1.0.1.0 ([Doc ID 1675966.1](https://support.oracle.com/epmos/faces/DocContentDisplay?id=1675966.1)) So you might have trouble accessing legacy systems, without reverting to HTTP.

For more information see:

* [orapki Utility](http://docs.oracle.com/cd/E11882_01/network.112/e40393/asoappf.htm)
* [UTL\_HTTP](http://docs.oracle.com/cd/E11882_01/appdev.112/e40758/u_http.htm)
* [Master Note For SSL/TLS (Doc ID 2229775.1)](https://support.oracle.com/epmos/faces/DocContentDisplay?id=2229775.1)