WSDL2RPG – FAQ

Proxy and HTTP Server Logon

Status of this document

Date: 15.09.2015
Version: 1.4

Question

How to specify user ID and password if a proxy or HTTP server requires authentication?

Answer

There are two situations when this problem may arise. The first situation is when you try to run WSDL2RPG to generate the web service client stub module using an HTTP URL. In that case WSDL2RPG will bring up a window to let you enter the user ID and password for the server.

The second situation where you may need to log on to a server is when you try to run the compiled web service stub module. In that case you have to register a callback procedure to provide the correct user credentials.

In order to register the callback procedure you have to call the `YourWebServicePort_Port_registerLoginCallback()` procedure. The procedure takes the procedure pointer of the callback procedure as an argument.

Prototype of Callback Procedure

```
 *
D YourWebServicePort_supplyLoginData...
D                PR            10I 0
D                                     extproc('+
D                                     YourWebServicePort+
D                                     _supplyLoginData+
D                                     ')
D  i_authType                  14A           const  varying
D  i_realm                    126A           const  varying
D  i_numAttempts               10I 0         const
D  o_user                      50A                  varying
D  o_password                  50A                  varying
 *
```

Description of Parameters and Return Value

Parameter	Description

`i_authType`	Specifies the type of server that request authentication.
	`WSDL_AUTH_TYPE_PROXY    Proxy Server`
	`WSDL_AUTH_TYPE_HTTP     HTTP Server`

`i_realm`	The realm used by the server to define access rules, such as user Ids and passwords.

`i_numAttempts`	Number of login attempts of the specified authentication type.
`o_user`	User ID to use.
`o_password`	Password to use.

Return value	**Description**
`return code`	Returns `WSDL_SUCCESS` on success, else `WSDL_ERROR`.

In case of a user specific account at the server, `o_user` may be set to the current user of the job. Then `i_authType`, `i_realm` and `o_password` may be used to get the password from a database table.

In case of an anonymous log-in only `i_authType` and `i_realm` may be used to get the password from a database table.

Default Login Procedure

The generated default `*_Port_supplyLoginData()` procedure keeps track of the number of login attempts and sends message "`Failed to get login credentials. Please customize procedure: *_Port_supplyLoginData`" (USR0048).

So in case that you receive a USR0048 error message, you should customize the default login procedure or use `YourWebServicePort_Port_registerLoginCallback()` to register your own login procedure.

Program Flow

The generated stub module calls `*_Port_login()`, when the web service requires authentication. `*_Port_login()` does some internal stuff and eventually calls the registered login procedure, which must implement the procedure interface as shown above (`YourWebServicePort_supplyLoginData`). An example of such a procedure is given below (`Namespaces_supplyLoginData`).

The interface of the login callback procedure is quite generic and considered to be stable for all WSDL2RPG versions.

From my point of view the best idea is to write your own service program that reads the login credentials from a database table. This way you get a central point of control for your login data that you can use for all your current and upcoming web services. Then use `YourWebServicePort_Port_registerLoginCallback()` to register your login procedure for a specific web service.

Hint

In case you want the current user to specify the user ID and password to log on, you may want to register `WSDL2R42_getLoginData()` as your callback procedure. It is shipped with WSDL2RPG and prompts the window that you may already know from the WSDL2RPG command.

```
YourWebServicePort_Port_registerLoginCallback(
      %paddr('WSDL2R42_getLoginData'));
```

Sample: General Information

The following sample was taken from one of my web service that I use for testing WSDL2RPG. The name of the port of the web service is “Namespaces”. Hence “Namespaces_” is the prefix of all procedures exported by the generated stub module.

Sample: Registration of Callback Procedure

```
         Namespaces_Port_registerLoginCallback(
               %paddr('Namespaces_supplyLoginData'));
```

Sample: Callback Procedure

```
      *
      * ==================================================================
      *  Procedure to supply login data if
      *  web server requires user authentication.
      * ==================================================================
      *  Returns WSDL_SUCCESS if successful, WSDL_ERROR upon error
      * ==================================================================
     P Namespaces_supplyLoginData...
     P
      *
     D Namespaces_supplyLoginData...
     D                 PI            10I 0
     D  i_authType                   14A          const  varying
     D  i_realm                     126A          const  varying
     D  i_numAttempts                10I 0        const
     D  o_user                       50A                 varying
     D  o_password                   50A                 varying
      *
      *  Return value
     D rc              S             10I 0
      * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      /free

         if (i_numAttempts > 1);
            return WSDL_ERROR;
         endif;

         if (i_authType = WSDL_AUTH_TYPE_PROXY);
            o_user     = 'proxyUser';
            o_password = 'proxyPassword';
            return WSDL_SUCCESS;
         endif;

         if (i_authType = WSDL_AUTH_TYPE_HTTP);
            o_user     = 'httpUser';
            o_password = 'httpPassword';
            return WSDL_SUCCESS;
         endif;

         return WSDL_ERROR;

      /end-free
      *
     P Namespaces_supplyLoginData...
     P                 E
```

Please notice, that the following statements have been removed to avoid an escape message at runtime:

```
clear USR0048;
USR00481 = f_this();
AppMsg_sendCancelMsg(AppMsg_newError('USR0048': USR0048));
```

It is sufficient to return `WSDL_SUCCESS` or `WSDL_ERROR` for a proper operation of the web serve stub.

The example procedure uses `i_numAttempts` to keep track of the number of login attempts. I strongly recommend to keep track of the number of login attempts if the login credentials are hard coded or read from a database table. Otherwise the program may enter an endless loop when the supplied credentials (e.g. the password) are wrong.

Instead of hard coding the user and password you may decide to read the login credentials from a database table.

Your comments are important to me! Please send me your comments about this FAQ. I will greatly appreciated it.

thomas.raddatz@tools400.de