WSDL2RPG - FAQ

How to Save Response to Stream File

Status of this document

Date: 18.09.2015
Version: 1.1

Question

How can I save the response of a web service call to a stream file in order to process the response by a different program later on?

Answer

Starting with WSDL2RPG v1.15.2 you can use the new `*_RedirectResponse_toStmf()` procedure to write the web service response message to a stream file. Prior to v1.15.2 you need to modify the generated web service stub by hand.

Using RedirectResponse_toStmf()

The good news here is that you do not need to tweak anything. Just call `*_RedirectResponse_toStmf()` right before invoking the web service procedure and you are done:

```
// Redirect response to stream file
if (stmf <> '');
   ZipCodeSoap_RedirectResponse_toStmf(stmf);
endif;

// Now let's call the web service.
CityStateToZipCodeResponse =
   ZipCodeSoap_CityStateToZipCode(
       parameters:
       errText);
```

The parameters required by procedure `*_RedirectResponse_toStmf()` are:

i_stmf	Required. Name of the stream file.
i_replace	Optional. Specifies whether or not an existing stream file is replaced.
	`cTrue | *ON` An existing file is replaced
	`cFalse | *OFF` Additional data is appended and delimited by CRLF
i_ccsid	Optional. CCSID that is assigned to the stream file when the file is created. Usually that is UTF-8 (= 1208).
i_auth	Optional. Authority that is assigned to the stream file when the file is created. The authority value is a composition of the following bit values (see parameter *mode* of the *chmod()* Unix type API):

Owner Authority

S_IRUSR	Read permission for the file owner
S_IWUSR	Write permission for the file owner
S_IXUSR	Search permission (for a directory) or execute permission (for a file) for the file owner
S_IRWXU	Read, write, and search or execute for the file owner. S_IRWXU is the bitwise inclusive OR of S_IRUSR, S_IWUSR, and S_IXUSR

Group Authority

S_IRGRP	Read permission for the file's group
S_IWGRP	Write permission for the file's group
S_IXGRP	Search permission (for a directory) or execute permission (for a file) for the file's group
S_IRWXG	Read, write, and search or execute permission for the file's group. S_IRWXG is the bitwise inclusive OR of S_IRGRP, S_IWGRP, and S_IXGRP

Public authority

S_IROTH	General read permission
S_IWOTH	General write permission
S_IXOTH	General search permission (for a directory) or general execute permission (for a file)
S_IRWXO	General read, write, and search or execute permission. S_IRWXO is the bitwise inclusive OR of S_IROTH, S_IWOTH, and S_IXOTH

Modifying the Stub by Hand

If you cannot use or do not want to update to v1.15.2, you need to tweak the WS_OPER module and the program that calls the web service procedure. The following description is just an example to show the general approach. You may want to vary this approach to better meet your needs.

Overview

The program calling the web service procedure is changed to open a stream file and to pass the file descriptor of the stream file to the web service procedure. It also closes the stream file after having called the web service.

The WS_OPER module is changed to receive the file descriptor and to write the web service response to the stream file. See example FQ0001.

The Program calling the Web Service Operation

First you need to add a field for the file descriptor of the stream file:

```
*
 *  Dynamic array index fields
D X_A1            S             10I 0 inz
 *
 *  Redirect response to stream file
D fd              S             10I 0 inz
```

Then you can open the stream file right before calling the web service procedure. Option `O_EXCL` ensures that the file must not yet exist. Also add parameter `fd` to the parameter list:

```
// Open stream file to take the response message.
fd = open(filename
          : O_CREAT + O_SHARE_RDONLY + O_WRONLY + O_CCSID + O_EXCL
          : S_IRWXU + S_IRWXG + S_IRWXO
          : 1208);

// Now let's call the web service.
CityStateToZipCodeResponse =
   ZipCodeSoap_CityStateToZipCode(
       fd:
       parameters:
       errText);

if (fd = -1);
   sndMsg('Failed to open stream file. File may already exist.');
   *inlr = *on;
   return;
endif;
```

Last but not least remember to close the file after having called the web service:

```
// Close stream file.
callp close(fd);
```

The WS_OPER Module

First the file descriptor has to be added to the prototype and the procedure interface of the WS_OPER module. Changes are coloured in red:

```
D ZipCodeSoap_CityStateToZipCode...
D                 PI                  likeds(s0_CityStateToZipCodeRespon...
D                                     se_t)
D  i_fd                         10I 0 const
D  i_s0_CityStateToZipCode...
D                                     likeds(s0_CityStateToZipCode_t)
D  o_msg                       128A   varying
```

Change the prototype accordingly.

Next you have to add a few fields to the web service procedure. In this example the name of the web service procedure is `ZipCodeSoap_CityStateToZipCode`:

```
 *
 *  User data
D userData        DS                  likeds(s0_CityStateToZipCodeRespon...
D                                     se_t)
D                                     inz
 *
 *  Redirect response to stream file
D pWriteProc      S                   *   inz procptr
D writeFD         S                 10I 0 inz
D isRedirectResp  S                   N   inz(cFalse)
```

pWriteProc	Takes the procedure pointer of the procedure, the web service response is forwarded to.
writeFD	File descriptor of the stream file.
isRedirectResp	Indicates whether or not the response is redirected to the stream file.

Then you need to decide whether or not the response most be redirected to the stream file. This example assumes that a file descriptor of -1 indicates that the response shall be processed by eXpat. A value different from -1 is assumed to be a valid file descriptor and hence redirection is enabled:

```
o_msg = '';

// Redirect web service response to stream file
if (i_fd <> -1);
   pWriteProc = %paddr('write');
   writeFD = i_fd;
   isRedirectResp = cTrue;
else;
   pWriteProc = %paddr('HTTP_receiveResponse');
   writeFD = 0;              // Be save, keep the original value
   isRedirectResp = cFalse; // Indicate that there is no redirection
endif;

monitor;
```

The next step is to change the statement that calls `http_url_post_raw2()` to use `writeFD` and `pWriteProc` instead of the hardcoded values:

```
rc = http_url_post_raw2(
   url
   : hInpStream
   : %paddr(
        'WSDL2R87_ManagedMemoryDataSource_InputStream_read')
   : ManagedMemoryDataSource_getSize(hDataSource)
   : writeFD
   : pWriteProc
   : getTimeout()
   : getUserAgent()
   : '');
```

Last but not least you need to ensure that only the real message is written to the stream file but no other information such as a “Not authenticated” error page. Add the following statement right after `http_url_post_raw2()`:

```
Rc = http_url_post_raw2(...)

// Truncate response stream file in case of an error
if (isRedirectResp and rc <> 1);
   ftruncate(writeFD: 0);
endif;

if (rc = 301 or rc = 302 or rc = 303 or rc = 307);
```

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

thomas.raddatz@tools400.de