WSDL2RPG – FAQ

FAQ How to create a Test Program

Status of this Document

Date:	31.01.2012
Version:	1.6

Question

What do I have to do to create a test program to call a Web Service?

Answer

In order to call a Web Service you have to do the following steps:

a) Let WSDL2RPG generate a Web Service stub for you.

b) Compile the base and stub module.

c) Link the stub module along with the base module to a service program.

d) Write a module for a little test program.

e) Link the test program and let it bind to the Web Service stub service program.

Generating a Web Service stub module

Prompt the WSDL2RPG command and specify the URL of your Web Service WSDL file as well as the output source member you want to use. You do not necessarily need to specify the name of the port and the name of the operation. Just keep the default values and WSDL2RPG will show you a screen with all available operations. Now select the desired operation and press ENTER.

```
WSDL2RPG   URL('http://theServer/TheRemoteWsdlFile.wsdl')
           SERVICE(*SELECT *SELECT)
           SRCFILE(yourLib/yourSrcFile)
           SRCMBR(WS0001) TYPE(*STUB)
```

or

```
WSDL2RPG   URL('file:/thePath/TheLocalWsdlFile.wsdl')
           SERVICE(*SELECT *SELECT)
           SRCFILE(yourLib/yourSrcFile)
           SRCMBR(WS0001) TYPE(*STUB)
```

You got: Two source members. Member **WS0001** is the base member that contains all procedures that are shared between all operations of the Web Service. Member **WS000101** the stub member which contains the RPG procedure that acts as a proxy to your Web Service operation.

Note: Please note the red "**WS0001**" that I used as the base name for the generated source members.

Compiling the generated modules

Use CRTRPGMOD to compile both source members. There are no special compiler settings necessary. However you should set DBGVIEW(*LIST) to be able to debug the program.

```
CRTRPGMOD  MODULE(yourLib/WS0001)
           SRCFILE(yourLib/yourSrcFile) SRCMBR(WS0001)

CRTRPGMOD  MODULE(yourLib/WS000101)
           SRCFILE(yourLib/yourSrcFile) SRCMBR(WS000101)
```

You got: Two modules that you can be link to a service program.

Linking the modules to a service program

Now link the base module and the stub module to a service program using the following parameter values for the CRTSRVPGM command:

```
CRTSRVPGM  SRVPGM(yourLib/WS0001)
           MODULE(yourLib/WS0001
                  yourLib/WS000101)
           EXPORT(*ALL)
           BNDSRVPGM(
              *LIBL/WSDL2RPGRT
              *LIBL/MIME
              *LIBL/HTTPMIME
              *LIBL/BASICS1)
           BNDDIR(QC2LE)
           TEXT('My first Web Service')
```

You got: A service program that exports the procedure you need to call the Web Service. You can consider that service program as a proxy for the Web Service.

Writing a test driver

Your test program must include the generated stub module (WS000101) as a copy book in order to know the prototype of the Web Service procedure and the type definitions of the request and response messages. Actually that should not work because the generated stub module contains executable code as well. In order to make it work you need to specify a compiler condition as shown below:

```
/DEFINE PROTOTYPE_WS000101
/COPY yourSrvFile,WS000101
/UNDEFINE PROTOTYPE_WS000101
```

That compiler condition ensures that the compiler includes only the prototype and the type definitions but no executable code.

Feel free to look at WS000101 (Web Service stub) and WS000101T (test program) to get a better idea of what I am talking about.

Now compile the module of your test program using a standard CRTRPGMOD command. Let us assume its name is WS000101T:

```
CRTRPGMOD  MODULE(yourLib/WS000101T)
           SRCFILE(yourLib/yourSrcFile) SRCMBR(WS000101T)
```

You got: A module object that you can link to a test program.

Linking the test program

The last step you have to do is to link your test program. The command you need is CRTPGM. The following parameters settings are required to create the test program:

```
        CRTPGM     PGM(yourLib/WS000101T)
                   MODULE(yourLib/WS000101T)
                   BNDSRVPGM(*LIBL/WS0001) BNDDIR(QC2LE)
                   TEXT('Test driver of my first Web Service')
```

In case your stub has been generated with DIM(*NOMAX) you need to add service programs **BASICS1** and **WSDL2RPGRT** to the BNDSRVPGM parameter of the CRTPGM command. (See sample: **WS0004**/**WS000401T**)

Sample

You may already have noticed the red and blue marked strings **WS0001, WS000101** and **WS000101T**. These things are a Web Service stub and its related test program. The Web Service behind the scene is hosted by Ripe Development LLC at http://www.ripedevelopment.com/. It provides lots of operations that all deal with zip codes, area codes and cities. The operation I selected is `CityStateToZipCode()`. It returns all zip codes for a given state and city.

Program Generator

Starting with v1.8, WSDL2RPG can generate a test program for you. That program is not very fancy and as simple as it can be. In order to generate the test program prompt the WSDL2RPG command and specify everything as usually, except for SRCMBR(). Change this parameter to the name of your test program. Then change the new TYPE parameter from its default (*STUB) to *PGM and press the ENTER key to displayed the STUB parameter. Set this parameter to the name of the stub module that you want to generate a test program for. Press ENTER again and let WSDL2RPG generate the program for you. The following command generates a test program for the **WS000101** stub module that we talked about before:

```
        WSDL2RPG   URL('http://www.ripedevelopment.com/webservices/ZipCode.asmx?WSDL')
                   SERVICE('ZipCodeSoap' 'CityStateToZipCode')
                   SRCFILE(*LIBL/QWSDL2RPG)
                   SRCMBR(WS000101T) TYPE(*PGM) STUB(WS000101)
```

Before you can compile and link the test program you first have to specify the request parameters of the Web Service. Open the source member with an editor of your choice and specify the request parameters right before the procedure that calls the Web Service. It should not be difficult to locate that part of the program. For example use the following parameter values for “Boston”:

parameters.City	:	Boston
parameters.State	:	MA

Now let us generate the stub modules:

```
        WSDL2RPG   URL('http://www.ripedevelopment.com/webservices/ZipCode.asmx?WSDL')
                   SERVICE('ZipCodeSoap' ('CityStateToZipCode'))
                   SRCFILE(*LIBL/QWSDL2RPG)
                   SRCMBR(WS0001) TYPE(*STUB)
```

When you are done create a service program from the base and stub module and a program from the module that you have just generated. The necessary commands are:

```
CRTRPGMOD  MODULE(WSDL2RPG/WS0001)
           SRCFILE(*LIBL/QWSDL2RPG) SRCMBR(WS0001)
           TRUNCNBR(*NO) DBGVIEW(*LIST)

CRTRPGMOD  MODULE(WSDL2RPG/WS000101)
           SRCFILE(*LIBL/QWSDL2RPG) SRCMBR(WS000101)
           TRUNCNBR(*NO) DBGVIEW(*LIST)

CRTSRVPGM  SRVPGM(WSDL2RPG/WS0001) MODULE(*LIBL/WS0001 *LIBL/WS000101)
           EXPORT(*ALL) TEXT('Web Service Stub')
           BNDSRVPGM(*LIBL/WSDL2RPGRT *LIBL/MIME
              *LIBL/HTTPMIME *LIBL/BASICS1) BNDDIR(QC2LE)

CRTRPGMOD  MODULE(WSDL2RPG/WS000101T)
           SRCFILE(*LIBL/QWSDL2RPG) SRCMBR(WS000101T)
           TRUNCNBR(*NO) DBGVIEW(*LIST)

CRTPGM     PGM(WSDL2RPG/WS000101T)
           MODULE(*LIBL/WS000101T)
           BNDSRVPGM(*LIBL/WS0001) ACTGRP(*NEW) BNDDIR(QC2LE)
           TEXT('Web Service Stub - Test Program')
```

Activation Groups

Did you notice that I specify `ACTGRP(*NEW)` for the sample program? I do that because this way everything is cleaned up when the sample program ends. When the sample program is called it is automatically loaded into a temporary activation group which is created on the fly by the operation system. The service program also goes into that group because of ACTGRP(*CALLER) which is the default value of `CRTSRVPGM`.

In case you want to use your Web Service stub from a RPG III program you have to write a RPG IV wrapper program used as the glue between these two worlds. Feel free to use the sample program as your wrapper program. But be careful and <u>never</u> change the activation group of the wrapper program to *CALLER! If you do that, the program will be loaded into the default activation group as well as the Web Service stub service program. IBM highly recommends avoiding that because you will never get rid of that service program until the job ends. No `RCLRSC` or RCLACTGRP will ever remove the service program from memory.

For example we had a problem with service programs using a file. When we put such a service program into the default activation group and did a `RCLRSC` the file was closed but the service program did not know about that. %open() always returned *ON although the file was closed.

Either specify `ACTGRP(*NEW)` which last in slower performance or a named activation group such as `ACTGRP(WEBSERVICE)` for your wrapper program. Because of `ACTGRP(*CALLER)` the service program will also be loaded into activation group WEBSERVICE.

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

thomas.raddatz@tools400.de