WSDL2RPG – FAQ

FAQ How to Upload Base64 Encoded Files

Status of this Document

Date: 03.10.2011
Version: 1.1

Question

My WSDL file specifies an element of type “base64Binary” in order to upload a stream file onto the server. How can I encode the file and add the encoded data to the request message?

Short Answer

Starting with WSDL2RPG V1.13 you can use the following procedure to encode a given stream file to Base64:

```
Marshaller_toBase64BinaryStream()
```

Answer

In order to utilize the new `Marshaller_toBase64BinaryStream()` procedure you first have to generate your stub module as usual and then modify it by hand.

Sample Web Service

The following explanation bases on method '`uploadBase64EncodedFile`' of the '`Base64FileService`' sample service. The WSDL file of that web service specifies the following request message:

```
<xsd:element name="uploadBase64EncodedFile">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="fileName" type="xsd:string"/>
            <xsd:element name="fileData" type="xsd:base64Binary"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Element `fileName` is intended to take the name of the file that is going to be uploaded. Whereas `fileData` is supposed to carry the Base64 encoded data.

Let us assume that `fileName` must contain the name of the file without path and that the file you want to upload is located at:

```
c:\temp\HelloWorld.txt
```

The content of the file might be the following simple and well know text:

```
Hello World!
```

Putting it all together the expected request message looks like that:

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="http://tools400.de/wsdl2rpg/webservice/sample/base64file/
service">
<soapenv:Header/>
    <soapenv:Body>
        <ser:uploadBase64EncodedFile>
    <fileName>HelloWorld.txt</fileName>
    <fileData>SGVsbG8gV29ybGQh</fileData>
        </ser:uploadBase64EncodedFile>
    </soapenv:Body>
</soapenv:Envelope>
```

Generating the Stub Module and the Test Program

Now let us jump in and generate the stub module as usual:

```
WSDL2RPG
    URL('http://tools400.dyndns.org:88/axis2/services/Base64FileService?wsdl')
    SERVICE('Base64FileService' ('uploadBase64EncodedFile'))
    SRCFILE(*LIBL/QWSDL2RPG) SRCMBR(WS0006)
```

Let us also generate a test program and set the request parameters:

```
WSDL2RPG
    URL('http://tools400.dyndns.org:88/axis2/services/Base64FileService?wsdl')
    SERVICE('Base64FileService' ('uploadBase64EncodedFile'))
    SRCFILE(*LIBL/QWSDL2RPG) SRCMBR(WS000601T)
    TAPE(*PGM) STUB(WS000601)
```

You truly agree with me that there is no doubt about to put '`Hello World.txt`' into `fileName`:

```
requestMessage.fileName = 'Hello World.txt';
```

But now it is getting a bit strange because we will put the full path name into `fileData`. Do not get confused. Things will get clearer soon.

```
requestMessage.fileData = '\home\raddatz\wsdl2rpg\Hello World.txt';
```

Finally we are going to change the generated stub module. But first let us remember how the original code of **WS000601** looks like:

```
'<fileName>' +
Marshaller_toString(
   i_tns_uploadBase64EncodedFile.fileName
   ) +
'</fileName>' +
'<fileData>' +
Marshaller_toBase64Binary(
   i_tns_uploadBase64EncodedFile.fileData
   ) +
'</fileData>' +
```

If we called the program now we would get an error because of invalid Base64 data in `fileData`, wouldn't we? Let us fix that right now.

Modifying the Stub Module

What we are changing now is to append the request as produced so far to the output stream and then replace procedure `Marshaller_toBase64Binary()` with `Marshaller_toBase64BinaryStream()`:

```
    '<fileName>' +
    Marshaller_toString(
       i_tns_uploadBase64EncodedFile.fileName
       ) +
    '</fileName>' +
    '<fileData>';
    ManagedMemoryDataSource_OutputStream_write(
          hOutStream: %addr(request) + 2: %len(request));

    Marshaller_toBase64BinaryStream(
          i_tns_uploadBase64EncodedFile.fileData
          : hOutStream
          : %paddr(
            'WSDL2R87_ManagedMemoryDataSource_OutputStream_write'));

    request =
    '</fileData>' +
```

Please notice that we pass **`fileData`** to `Marshaller_toBase64BinaryStream()` to let it know the file that it has to encode.

Under the Cover

Under the cover `Marshaller_toBase64BinaryStream()` opens the file in binary mode and reads its data chunk by chunk. Each chunk is encoded to Base64 and appended to the output stream '`hOutStream`' by procedure `ManagedMemoryDataSource_OutputStream_write()`.

Actually that is all you need to know about it. However there may be one thing you should not forget:

Do not forget to call **`ManagedMemoryDataSource_OutputStream_write()`** before you encode your file. Otherwise the content of field '`request`' as been assigned to it so far will never be added to the output stream and hence never go over the wire.

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

thomas.raddatz@tools400.de