WSDL2RPG – FAQ

FAQ How to Send Base64 Encoded Passwords

Status of this Document

Date: 24.11.2011
Version: 1.0

Question

My WSDL file specifies an element of type “base64Binary” to carry a password to the server. Although I specify the proper password I get back an error message “Invalid User/Password”. What am I doing wrong?

Short Answer

In order to solve that problem you have to translate your password from EBCDIC to the CCSID of the target system before the Base64 marshaller encodes the password to Base64. You can include the necessary statements directly in the stub module:

```
 *  Temporary fields for converting the password
D passwordEbcdic  S            512A   inz
D passwordAscii   S                   like(passwordEbcdic )
D                                     based(pPasswordAscii)
D length          S             10I 0 inz

      // Translate password from job CCSID to
      // CCSID of the remote POST data
      passwordEbcdic = i_tns_AuthCredentials.Password;
      length = HTTP_xlatedyn(
                  %len(i_tns_AuthCredentials.Password)
                  : %addr(passwordEbcdic)
                  : TO_ASCII
                  : pPasswordAscii);

      // Produce SOAP request message
      request =
         '<?xml version="1.0" encoding="UTF-8"?>' +

      ...

      '<tns:Password>' +
      Marshaller_toBase64Binary(
         // i_tns_AuthCredentials.Password
         %subst(passwordAscii: 1: length)
         ) +
      '</tns:Password>' +

      ...

      // Do not forget to free the password pointer
      dealloc(N) pPasswordAscii;
```

Starting with v1.15 you can add getPostCcsid() as the last parameter to Marshaller_toBase64Binary() to let your stub module take care for the correct character conversion:

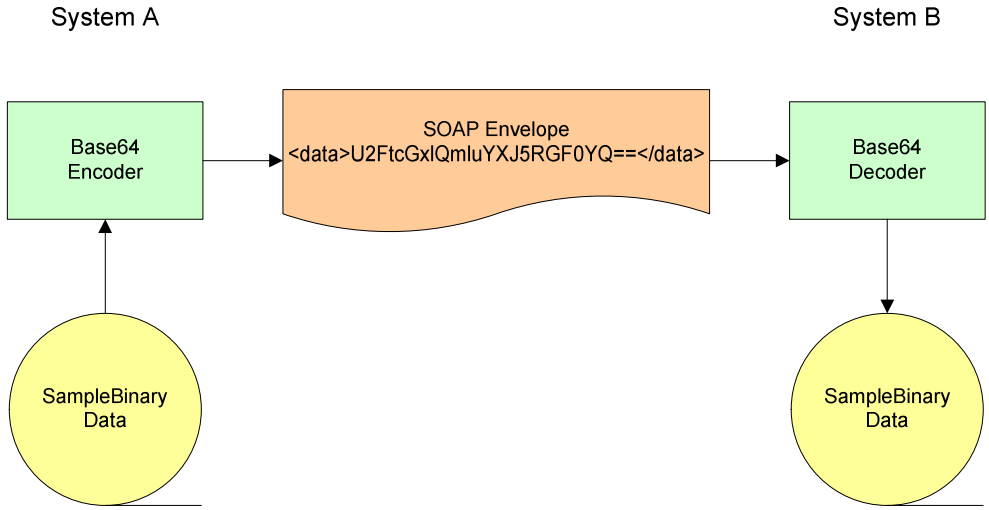
```
Marshaller_toBase64Binary(
      i_tns_AuthCredentials.Password
      : getPostCcsid())
```

Answer

Actually the Base64 encoding is used to send true binary data, for example a file, from one system to another system. Of course everybody expects that both files (source and target file) must exactly match. On the other hand people forget that basic feature of Base64 when sending passwords or other character data to a remote system. While this is not a problem if both systems use the same character encoding scheme, it is a problem if the CCSIDs are different. Most of the time that is true if our beloved “System i” is part of the communication.

Sending Binary Data

Again, if we are going to send binary data we want to receive the data unchanged on the target system. Hence usually the process is simple:



First System A reads the data from the file and passes it to the Base64 encoder. Then the encoded data is added to the SOAP envelope and send to System B.

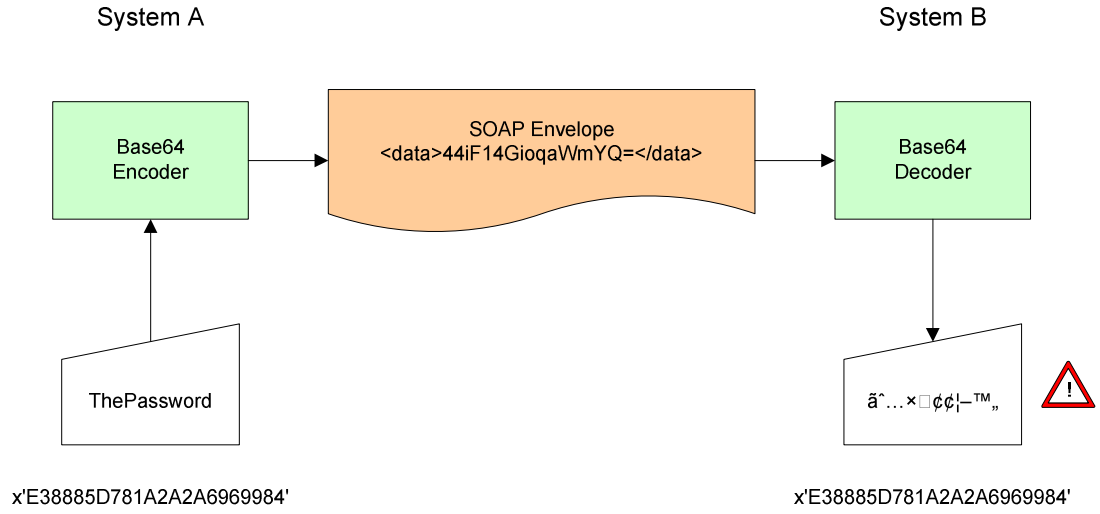
The target system retrieves the Base64 encoded data stream from the SOAP envelope and calls the Base64 decoder to decode the data. Eventually the decoded data is written to a file.

At the end of the process file B is an exact copy of file A.

Sending Character Data

Now let us see what the difference that makes for character data. Let us assume we have a System i that wants to exchange data with a Java application. The CCSIDs we are faced with are EBCDIC on our local system and UTF-8 on the remote server.

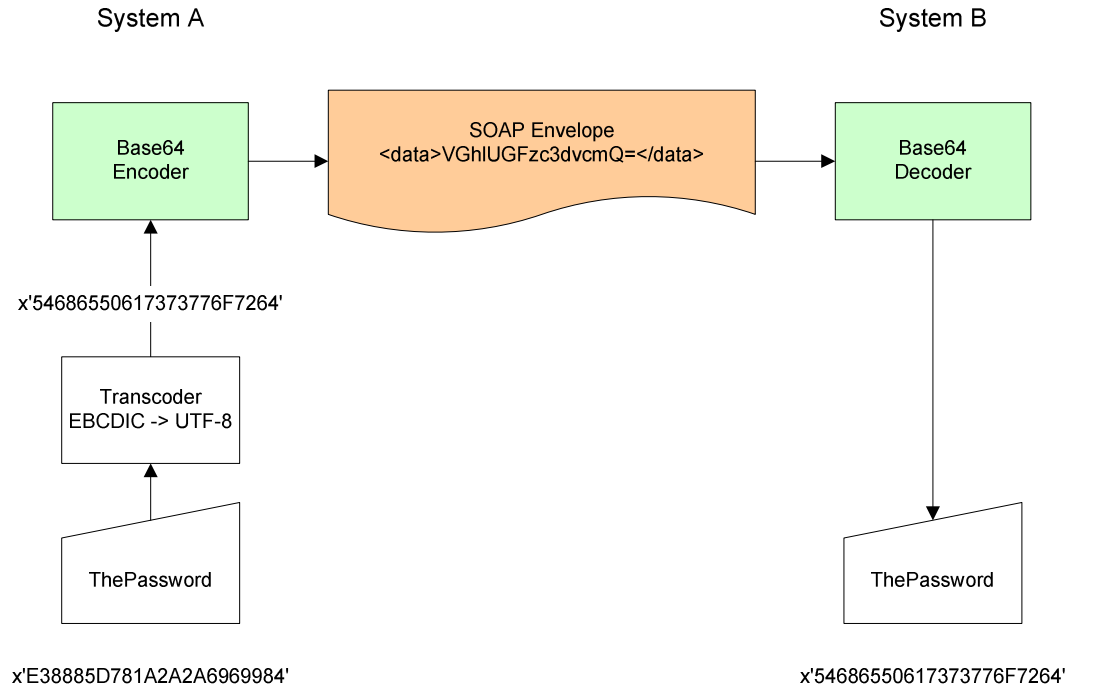
So what happens when we transfer the data just the way we do it for binary data? The following picture illustrates the answer:



Since the byte sequence of the character data is exactly the same on both systems, the character representation is different, because the source system used EBCDIC to represent “ThePassword” and the target system uses UTF-8 to interpret the byte sequence.

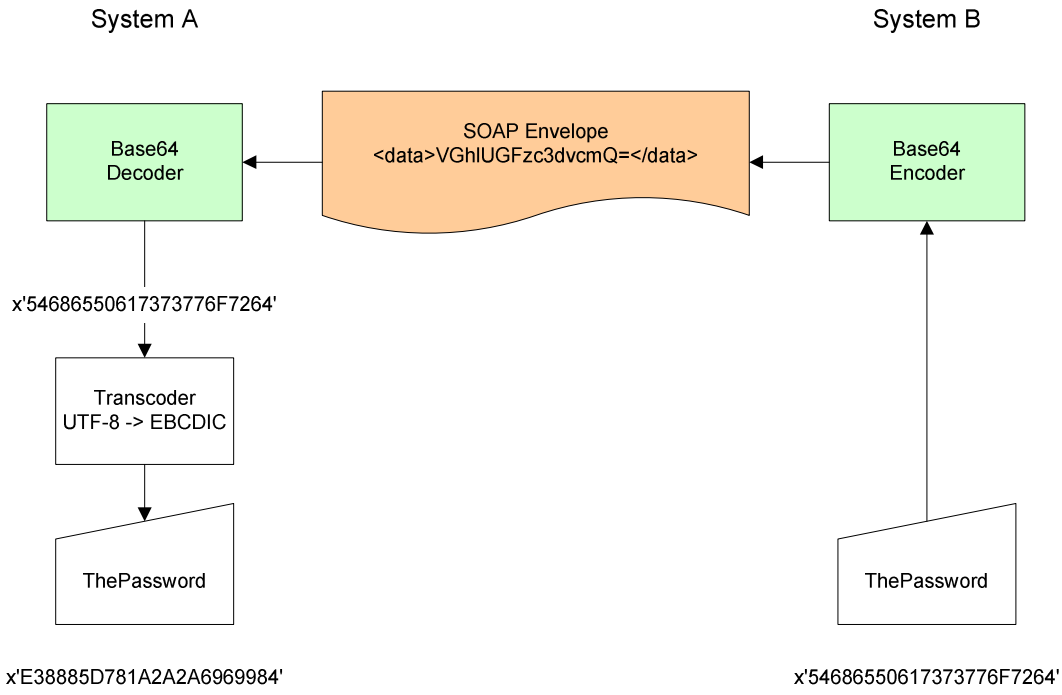
Naturally a following logon process must return “Invalid User/Password”.

We can fix that problem by translating “ThePassword” from EBCDIC to UTF-8 before passing it to the Base64 encoder as shown here:



Receiving Character Data

You may already know what to do when you receive Base64 encoded data from a UTF-8 system, don't you? The same character encoding problems we have when sending character data happens to happen when receiving character data. In order to fix that problem we have to convert the decoded data from UTF-8 to EBCDIC. Please notice the different data flow from right to left:



Integrated Character Translation Support

Starting with v1.15 WSDL2RPG can do the proper character translation for you. Either set the new parameter BASE64TYPE to *CHAR when generating the stub module or change Marshaller_toBase64Binary() and UnMarshaller_toBase64Binary() by hand as shown below:

Sending data:

```
Marshaller_toBase64Binary(
      i_tns_AuthCredentials.Password
      : getPostCcsid())
```

Receiving data:

```
currentItem.value =
   UnMarshaller_toBase64Binary(
          value
          : getPostCcsid())
```

Please remember that you have to add `getPostCcsid()` for character data and that you have to remove it for binary data. It is not unlikely that you may have stub modules with both "flavours" of the Base64 marshallers and unmarshallers depending on the type of data.

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

thomas.raddatz@tools400.de