Tip: SAS and Base64

Michael Dixon
4 years ago · Updated

Base64 is a method of encoding a file into normal printable characters so that it can be more easily transmitted electronically. This is particularly useful for binary files where the contents may contain special control, or other "non-printable", characters that could get lost in translation went sent electronically from A to B.

From WikiPedia:

"Base64 encoding schemes are commonly used when there is a need to encode binary data that needs to be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remains intact without modification during transport."

Base64 is in common use - some places you will find it are:

- » email attachments
- » binary data in XML documents
- » embedded images in HTML pages
- » code obfuscation

Base64 is also used by SAS - probably without you even knowing, e.g.:

- » SAS001 Password encoding (Shhh! don't tell anyone!)
- » Email attachments using the EMAIL engine of the FILENAME stateme
- » Numeric values with the XML LIBNAME engine when XMLDOUBLE=

Support

effect

To work directly with Base64 data in SAS you would (previously) need to take a look at the algorithms used to encode/decode streams of bytes and then implement those methods in SAS. You can find the information to do this in RFC 4648 if you would like to rebuild this from the ground up. There is also a SAS Global Forum 2011 paper by Erik Tilanus titled "The Trilogy on e-Mailing, Part 3: Handling e-Mail Attachments with SAS®", where the Base64 decoding method of manipulating 4 x 6-bit bytes back into 3 x 8-bit bytes is shown as implemented in the SAS Data Step language!

With the introduction of the BASE64Xw.d format/informat with SAS 9.2, this allows me to "cheat" a bit and create some simpler code to help work with Base64 data in SAS. While these new formats/informats let me work above the bit/byte-level manipulation required, they are limited to a maximum string length of 32k - which is fine for working with data field values, but probably not long enough if I want to encode a whole file. When working out how to break a longer string (or a stream of bytes from a file) down so that it can be handled by the provided format/informat, the trick here is:

- » For encoding, 3 characters = 4 characters
- » For decoding, 4 characters = 3 characters

So, all I have to do for encoding is read my source "stream of bytes" in multiples of 3 bytes/characters, and for decoding read in multiples of 4 bytes/characters.

Let's look at a data step that can encode a file of any size into Base64:

```
data _null_;
  length b64 $ 76 line $ 57;
  retain line "";
  /* Read one byte at a time from the input file */
  infile "C:\Temp\my_input_file.xls" recfm=F lrecl= 1 end=eof;
  input @1 stream $char1.;
  /* Write Base64 with line length of 76 characters */
  file "C:\Temp\my_output_file.txt" lrecl=76;
  /* Place the current byte into the line buffer */
  substr(line,(_N_-(CEIL(_N_/57)-1)*57),1) = byte(rank(stream));
  /* 57 source bytes get base64 encoded to 76 bytes */
  if mod(_N_,57)=0 or EOF then do;
    /* Convert the buffer to base64 */
    if eof then b64=put(trim(line),$base64X76.);
    else b64=put(line, $base64X76.);
    /* Write the base64 string out */
    put b64;
    /* Reset the buffer */
    line="";
  end;
run;
```

Hopefully the code is commented enough let you understand what is happening. I am basically reading in an Excel file one byte at a time until I have collected 57 bytes (57 is divisible by 3), then I apply the SAS BASE64Xw.d format and write it out to another (plain text) file where it the result is a 76 character string - (57 / 3) * 4 = 76.

Decoding is a bit easier, but pretty much a reverse of the above:

```
data _null_;
  length b64 $ 76 byte $ 1;
  /* We only deal with Base64 files with a max line length of 76 */
  infile "C:\Temp
\my_encoded_file.txt" lrecl= 76 truncover length=b64length;
  input @1 b64 $base64X76.;
  if _N_=1 then putlog "NOTE: Detected Base64 Line Length of
" b64length;
  file "C:\Temp\my_decoded_file.xls" recfm=F lrecl= 1;
  do i=1 to (b64length/4)*3;
    byte=byte(rank(substr(b64,i, 1)));
    put byte $char1.;
  end;
run;
```

Here we can use the SAS BASE64Xw.d informat directly on the INPUT statement to read in the lines of Base64 encoded data. The semi-tricky part is going through the resulting "string" byte by byte to output those bytes to a file exactly as they are - which is the DO loop getting the BYTE of each byte in the string before outputting that byte using the SAS $CHARw. format.

Probably the most useful routine out of the above is the encoding. One example of using this is to embed an image file directly into a HTML page - that way the image will always show up as it doesn't rely on a separate image file to be present.

```
data _null_;
 length b64 $ 76 line $ 57;
```

```
  retain line "";
  /* Read one byte at a time from the input file */
  infile 'http://www.scorpiosoftware.com.au/wp-content/themes/scorpio
/images/scorpio-logo.png'
     url recfm=F lrecl= 1 end=eof;
  input @1 stream $char1.;
  /* Write Base64 with line length of 76 characters */
  file "..\Temp\my_output_file.htm" lrecl=76;
  if _N_=1 then do;
    /* Write the beginning of the HTML doc */
    put '<html>';
    put '  <body>';
    put '    <p>Hi there!</p>';
    put '    <img src="data:image/png;base64,';
  end;
  /* Place the current byte into the line buffer */
   substr(line,(_N_-(CEIL(_N_/57)-1)*57),1) = byte(rank(stream));
  /* 57 source bytes get base64 encoded to 76 bytes */
  if mod(_N_,57)=0 or EOF then do;
    /* Convert the buffer to base64 */
    if eof then b64=put(trim(line),$base64X76.);
    else b64=put(line, $base64X76.);
    /* Write the base64 string out */
    put b64;
    /* Reset the buffer */
     line="";
  end;
  if eof then do;
    /* Close the image tag and write the end of the HTML doc */
    put '">';
    put '  </body>';
    put '</html>';
  end;
run;
```

The core reason that I ended up creating these routines was to upload files using SOAP via a web service... more on that in a later post!

What I ended up with was a single macro that I could call to either encode/decode a file, and would also take either a FILEREF or a FILENAME as parameters for both the input and output files (it works this out by itself). The source code to the macro is attached to this Article.

The syntax to this macro is:

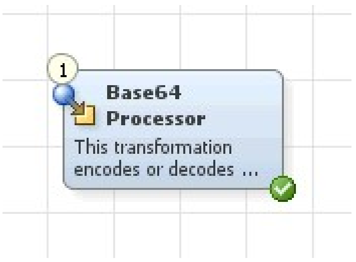
```
%base64(
  OPERATION=,   /* ENCODE or DECODE */
  FILEMODE=,    /* Create new/add to existing file */
  INPUT=,       /* Input FILEREF or FILENAME */
  OUTPUT=       /* Output FILEREF or FILENAME */

);
```

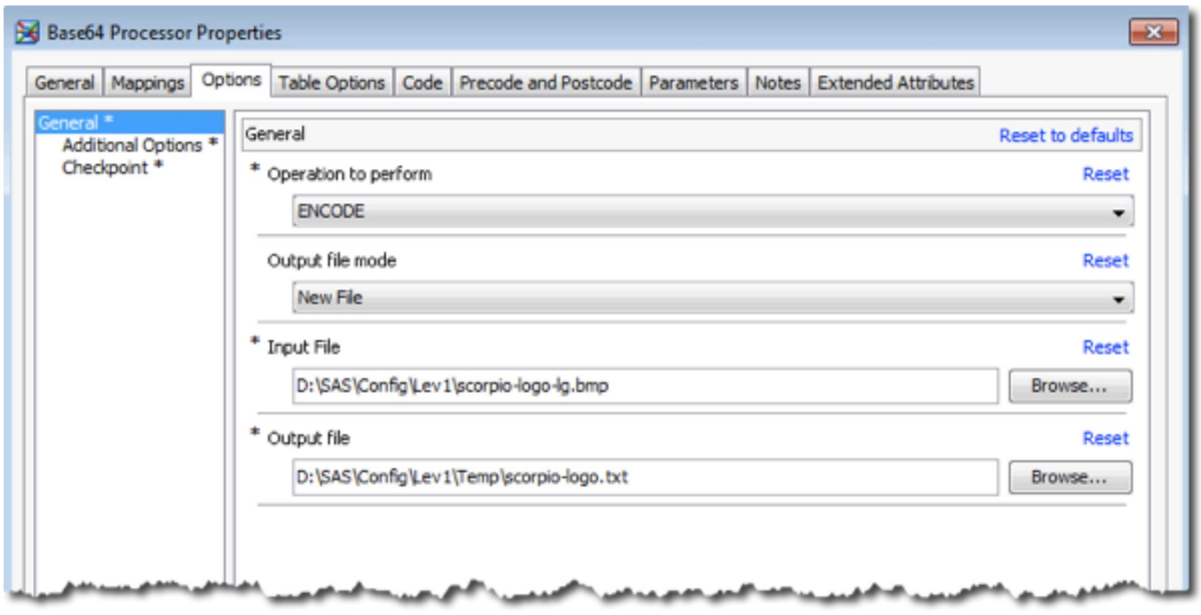
Utilising this macro as part of the previous example (embedding an image) would be:

```
data _null_;
  file "C:\Temp\myfile.html";
  put "<html>"
  put "  <body>";
  put '    <image src="data:image/png;base64,';
run;
%base64(OPERATION=ENCODE,INPUT=C:\Temp\logo.png,OUTPUT=C:
\Temp\myfile.html,FILEMODE=MOD);
data _null_ ;
  file "C:\Temp\myfile.html" mod;
  put '">';
  put "  </body>";
  put "</html>";
run;
```

I have also created a DI Studio Transformation so that I can use this functionality in my DI Studio jobs (SAS Package attached to this Article):



The transform provides a nice GUI for entering the parameters:


Base64 Processor Properties
General | Mappings | Options | Table Options | Code | Precode and Postcode | Parameters | Notes | Extended Attributes
General *
Additional Options *
Checkpoint *
General
Reset to defaults
Reset
Browse...

* Operation to perform: ENCODE

Output file mode: New File

* Input File: D:\SAS\Config\Lev1\scorpio-logo-lg.bmp

* Output file: D:\SAS\Config\Lev1\Temp\scorpio-logo.txt

Compatibility	
SAS 9.2	Yes
SAS 9.3	Yes

This article was originally posted 8May2012 at http://www.scorpiosoftware.com.au/2012/05/sas-and-base64/

Base64Transform.spk

7 KB · Download

base64.sas

3 KB · Download

2

Was this article helpful?

✓ Yes ✕ No

1 out of 1 found this helpful

Have more questions? Submit a request

Return to top

Related articles

Version Control for SAS DI Studio - CollabNet Subversion Edge

Tip: Platform Suite for SAS Dev/Test/Prod Tips

X11 and Java 7 Swing Apps (i.e. SAS Java Clients)

Tip: Adventures with DI Studio 4.3 - XML and Nicknames

SAS Log Error Checking Tool

Comments

2 comments

Sort by

Yves Payette

9 years ago

0

Is it possible to use your sas macro to decode a base64 blob stored in a MySQL variable into a txt file?

I have a whole table of blob variable to decode in multiple text files.

Thanks.

Michael Dixon

9 years ago

0

Hi Yves, The macro as described will only work with Base64 data in an external file, but with a slight tweak to the decoding routine above you could use the following code as a start:

```
data _null_;
length b64 $ 32767 byte $ 1; /* < -- Increase b64 length to max */
/* Change INFILE and INPUT statements */
set mysql.table_with_b64blobs;
b64=input(blob_var,$base64X32767.);
b64length=length(blob_var);
/* The rest is the same */
putlog "NOTE: Detected Base64 Line Length of " b64length;
file "D:\Temp\my_decoded_file.png" recfm=F lrecl= 1;
```

```
do i=1 to (b64length/4)*3;
byte=byte(rank(substr(b64,i, 1)));
put byte $char1.;
end;
run;
```

The caveat is that because the max string length that can be manipulated in SAS is 32k, your Base64 blob fields in MySQL have to contain 32k characters or less.