Paper DH12

The Encoding Dilemma

Angelo Tinazzi, Cytel Inc, Geneva, Switzerland

ABSTRACT

It is not uncommon to see in SAS log files messages such as "*Some character data was lost during transcoding in the dataset.*" What are we risking if we do not take care of such a message? What is the message telling us? How can we prevent the message to occur or what actions do we need to take to eventually correctly reading the data we have received?

Moreover, regulatory agencies such as the Japanese (PMDA) or more recently the Chinese (NMPA) [8], are now asking to submit datasets, and not surprisingly (or surprisingly for someone), they are requesting not only datasets in CDISC format but they are also requesting or suggesting the use of SAS XPT format as a file data format; furthermore, the NMPA has specific requirements with regards to the use of Chinese language for things like label of datasets and variables and contact of character variables e.g. adverse events terms.

With this paper I would like to stress the importance of encoding in SAS and how this could affect the correct handling of text. Options will be provided to make sure your character data are correctly retrieved when 'special' characters are handled. The risk of data loses when using SAS XPT and practical solutions will be also discussed.

INTRODUCTION

Very often SAS logs are plenty of let me say weird messages and often SAS programmers ignore these messages unless an error prevents their SAS session to execute correctly and without errors. I am sure most you have come across with at least one of the messages in figure 1.

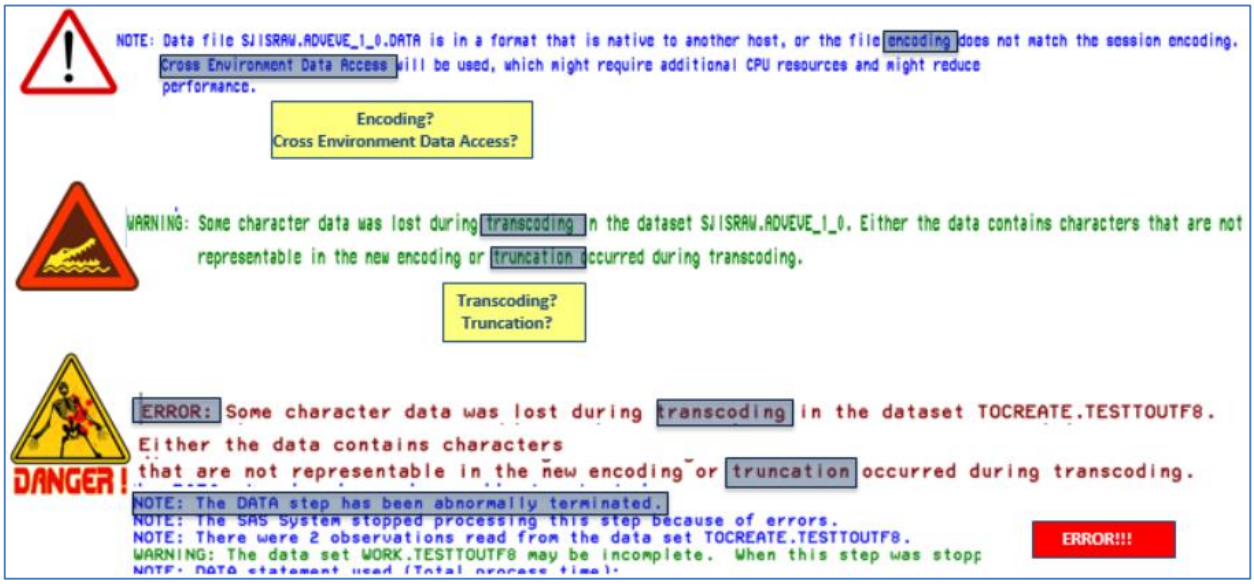

Figure 1: "Weird" messages in our SAS log

They are notes and warnings telling you that SAS has done some actions for you; while in the first case your local SAS session was able to correctly transcode characters coming from a different SAS encoding configuration, in the second case not all characters were correctly transformed and interpreted by SAS and even worst some characters make SAS to fail.

ENCODING IN A NUTSHELL

Before we try to troubleshoot and resolve these transcoding issues, it is helpful to understand why those issues occurred.

WHAT IS ENCODING

In computers characters data are stored as a series of bytes, which is made up of binary digits called bits. In this content a character is a generic wording to say any kind of character that could be represented by a computer system such as letters, digits or numbers, punctuations, and other special symbols.

As such in computers each character to be correctly represented is assigned a number. For example, with one byte, so eight bits, we can represent 256 characters, each one identified by a unique number ranging from 0 to 255.

This way computers interpret and represent the characters within the data by assigning an integer number, is called *encoding* and each encoding system has its own way of assigning a number to each character.

Thus, the *encoding method* of a specific encoding system is defined by a set of rules where each number is assigned a character. However, because our computing environments are not perfectly standardized, there are diverse types of encodings in use, and each encoding has its own character set and *encoding method*.

For example, in SAS with the wlatin1 encoding the letter "A" upper case is represented by the number "65" which is then translated into a sequence of eight bits (one byte). With this encoding system, we can represent most of the Western European Languages, such as English. We can represent all ASCII characters, including control characters and extended ASCII characters (figure 2).

Instead, other languages such as Cyrillic, or most of the Asian languages can be not fully or not represented at all because their alphabets use a separate set of symbols. Because of that, because some encodings require to represent different and more characters, the storage size of each encoding might differ.

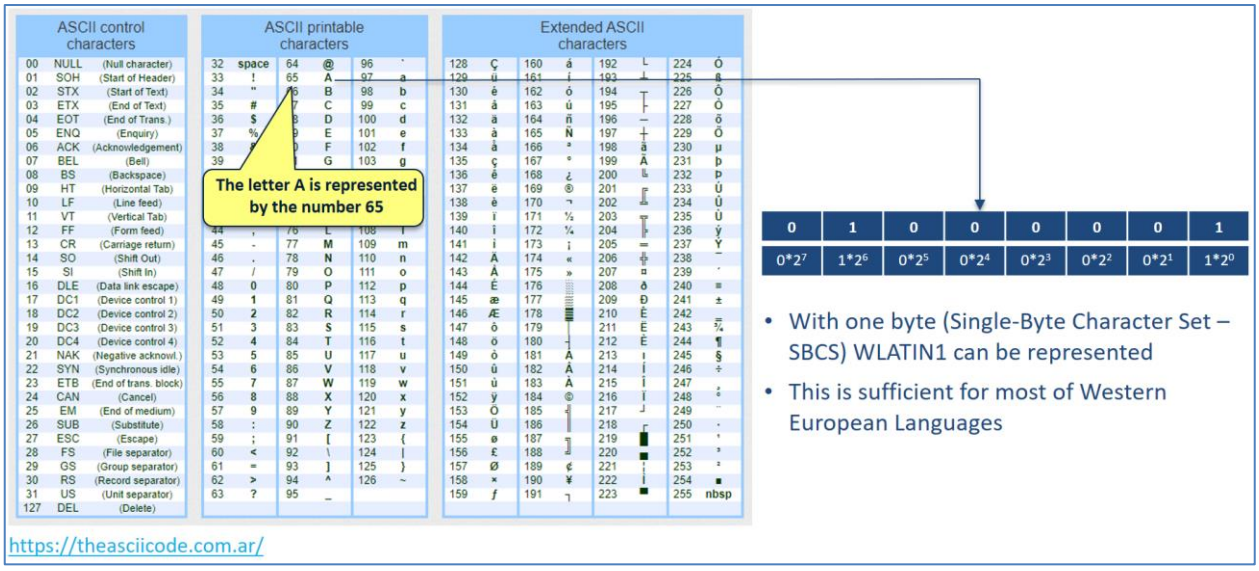

Figure 2: Encoding Method, example WLATIN1

SINGLE BYTE CHARACTER SET (SBCS) VS MULTIPLE BYTE CHARACTER SET (MBCS)

While with wlatin1 one byte is enough, so it is said it is a *Single Byte Character Set (SBCS)*, for other encoding one byte is not enough. For example, Shift-Jis the encoding system that defines encoding rules for Japanese characters and Big5 or EUC-UN for Chinese characters, require two bytes to represent all characters fully and correctly in their respective characters sets.

The UTF-8 encoding, defined by the Unicode standard, it is an attempt to overcome the different national languages requirements. With this encoding system in fact more than >120000 characters can be represented, some might still need one byte, other two, with up to a maximum of 4 bytes. For this reason, this encoding is defined as a *Multiple Byte Character Set* (MBCS). One also important thing to mention is that some characters that in wlatin1 require 1 byte, in UTF-8 the same.

Character Set	Main SAS encoding and characteristics
Single-Byte Character Set (SBCS)	• wlatin1 or wlatin9 for Western European • wlatin2 for Eastern European
Double-Byte Character Set (DBCS)	• Shift-Jis for Japanese • Big5 or EUC-CN for Chinese
Multiple-Byte Character Set (MBCS)	• UTF-8 Unicode – Universal Character set Transformation • An encoding that attempt to represent all characters in all languages MBCS (it uses 4 bytes) • >120000 characters. • For some 1 byte is still sufficient for many others not

Table 1: Type of Character Set

ENCODING AND SAS

The encoding defined by your SAS working environment, determines the default encoding for your SAS session.
For example, all your CDISC SDTM datasets created in your SAS session, are saved using the encoding defined by your SAS environment, so if your configuration use by default wlatin1 than by default any SAS datasets you created, either permanent or temporary, will be created using the wlatin1 encoding.
For this reason, it is important and critical to be aware of what encoding is by default used by your SAS configuration so that you can also inform and prevent your client or your regulatory agency.

DETECTING YOUR ENCODING

You can check this in SAS in at least two ways by querying the encoding option using the proc options, for example the default SAS configuration at Cytel uses wlatin1 encoding (figure 3)

```
                                        proc options option=encoding;
                                        run;

 ENCODING=WLATIN1  Specifies the default character-set encoding for the SAS session.
NOTE: PROCEDURE OPTIONS used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Figure 3: Encoding Method, example WLATIN1

ENCODING VS TRANSCODING

We discussed so far about encoding but let us also clarify a couple of topics related to transcoding.
Transcoding is the process of converting from one computer encoding to another. Giving the fact a character could have different numeric representations in two different encodings, if we are receiving datasets generated in a SAS environment with a different encoding, the first thing SAS will try to do is to make the needed transformations, or transcoding, so that each character is correctly represented and converted in the encoding used by our SAS configuration. SAS does automatic transcoding for you by invoking the *Cross-Environment Data Access* (CEDA) engine we saw also mentioned earlier in the SAS log.

DEALING WITH TRANSCODING IN SAS

We saw before how to check the encoding used by our SAS configuration, let's see how to check the encoding used in the SAS datasets received. SAS supplies several options and one is the **`ATTRC`** function that can be used with **`%sysfunc`** macro. In figure 4 you can see a simple macro I have created to make a number of checks of datasets encoding.

```
/*Checking encoding*/
%macro encod(libin=,dsin=);
  %put;
  %put ----CHECKING FOR ENCODING----------------------------;
  %put       &libin..&dsin;
  %let dsid=%sysfunc(open(&libin..&dsin,i));
  %put       ENCODING is %sysfunc(attrc(&dsid,encoding));
  %put ------------------------------------------------------;
  %put;
  %put;
  %let dsclose=%sysfunc(close(&dsid));
%mend;
```

Figure 4: Checking the dataset encoding – Macro %encod

For the purpose of the examples I'm going to discuss, I will make use of four libraries each one containing SAS datasets created with different encoding:

- **UTF8** a SAS library with SDTM datasets generated in an UTF-8 Unicode encoding environment
- **SJID** a SAS library with SDTM datasets generated in a shift-Jis Japanese encoding environment

- **WLATIN** a SAS library with SDTM datasets generated in WLatin1 western encoding environment
- **SJISRAW** a SAS library with legacy datasets generated in shift-Jis Japanese encoding environment. Some datasets contain some variables with Japanese characters

Figure 5 shows some example of using the macro to test the encoding of datasets created with different encoding:

- the first dataset queried is a dataset created using UTF-8 encoding that is different than my encoding
- same for the second dataset created with shift-Jis Japanese encoding. For both datasets SAS is alerting us the SAS datasets were generated using a different encoding
- for the last example, because the dataset was created with the same encoding, no NOTES are written in the log

Of note, the other way of checking the encoding is by using the **`PROC CONTENTS`** (figure 6).

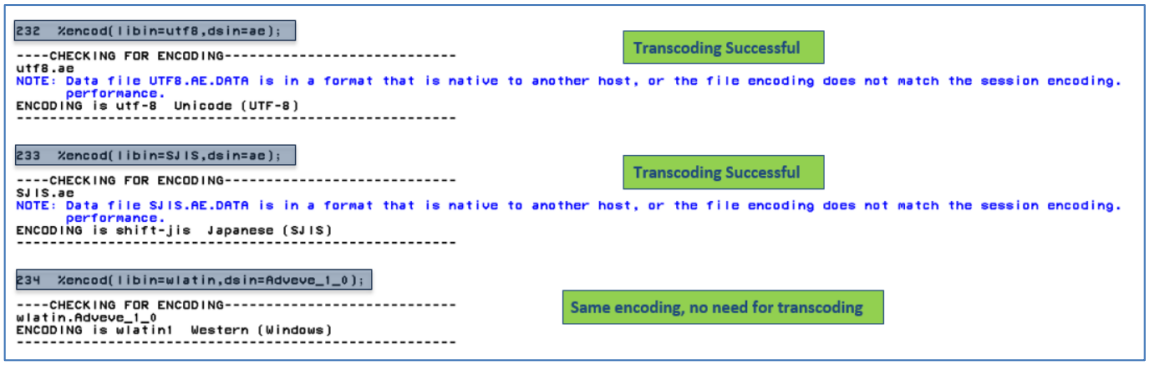

Figure 5: Checking the dataset encoding

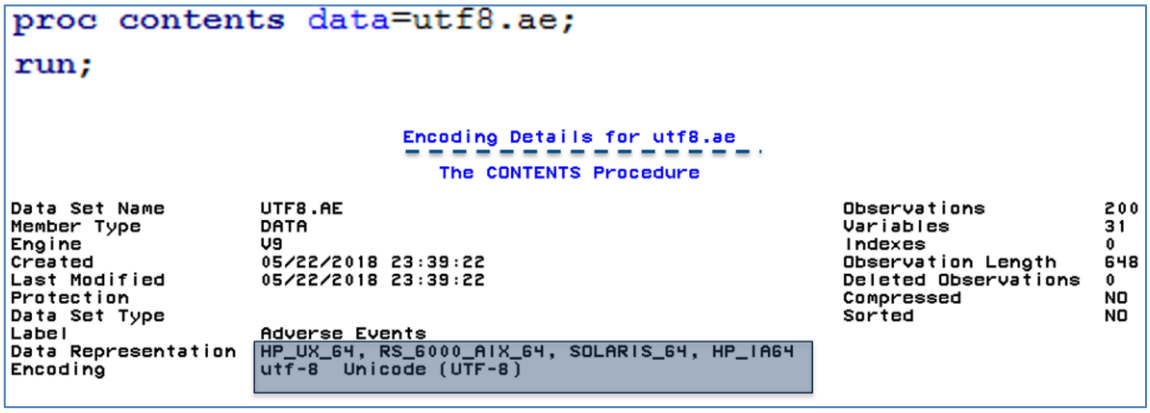

Figure 6: Checking the dataset encoding with the PROC CONTENTS

I mentioned before that, when a dataset is generated using a different encoding, SAS tries to make the transformation for you; however, there are situations where some characters can be not represented into your SAS encoding session and as such the characters are lost.

For example, despite our transcoding seemed successful when previously we queried the Japanese AE dataset, if we try now to open the same dataset, we will see that for one variable in particular some characters were not correctly translated and despite the Cross Environment Data Access tried to transcode, some characters in the Japanese adverse event verbatim text were lost (figure 7).

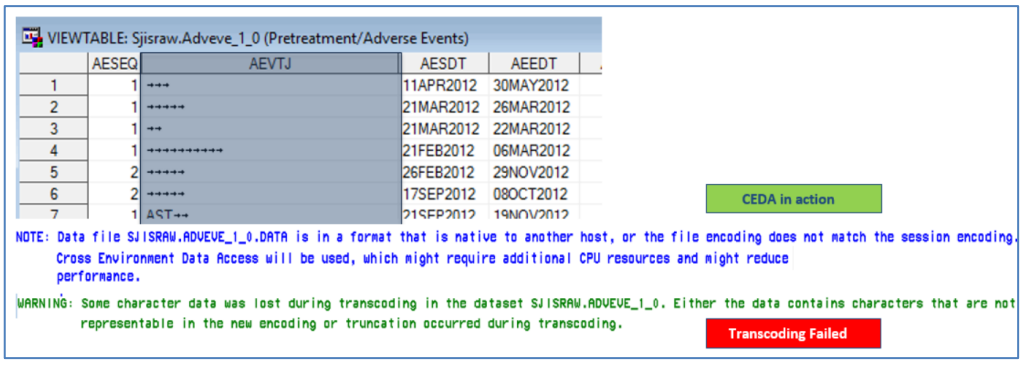

Figure 7: Opening a dataset with shift-Jis Japanese (SJIS) encoding with Japanese

In such a situation the only way of correctly reading the dataset content, it is to change the default encoding of your SAS configuration in your SAS configuration file. Let us assume we are not aware of the encoding used, we can try with the universal UTF-8:

```
-ENCODING UTF-8
```

So what before was partially read when the dataset was opened in SAS wlatin1 encoded session, it is now correctly read in a UTF-8 encoding SAS session (figure 8), of course if you can read Japanese!

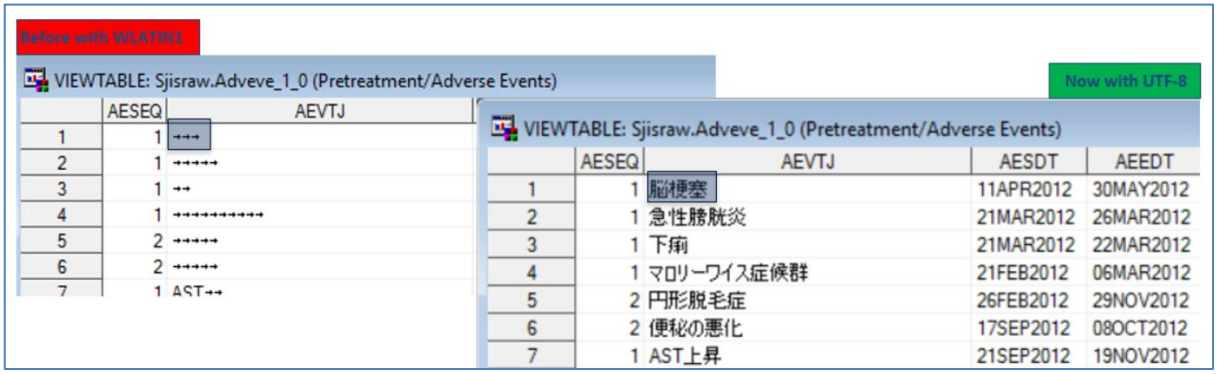

Figure 8: Checking the dataset encoding

From WLATIN1 to UTF-8

With the earlier example we saw that changing the encoding of our SAS session to match the source encoding could solve the issue. However, there are other instances where this is not enough.

Let us now take the example of lab dataset originally created with a wlatin1 session when this is transcoded into a UTF-8. See figure 9: despite the SAS Cross Environmental Data Access, one character was not correctly transcoded. I earlier mentioned that the same character in one encoding system might requires one byte, while when transformed onto another encoding the same could require two or more bytes. If we look for example at the 256 ASCII characters, we can see that the standard ASCII characters require 1 byte in UTF-8, same as in wlatin1, while the extended ASCII characters in UTF-8 requires two bytes where wlatin1 require one byte.

Thus, the “micro” (µ) symbol is on the list of characters that in UTF-8 requires two bytes instead of one and if we look at the “numeric” representation in the two encoding systems we can see they are two different “character”. This difference is also reflected in the SAS length, where the micro symbol is of length 1 in a wlatin1, while the same is of length 2 in UTF-8. Does this explain the wrong transcoding?

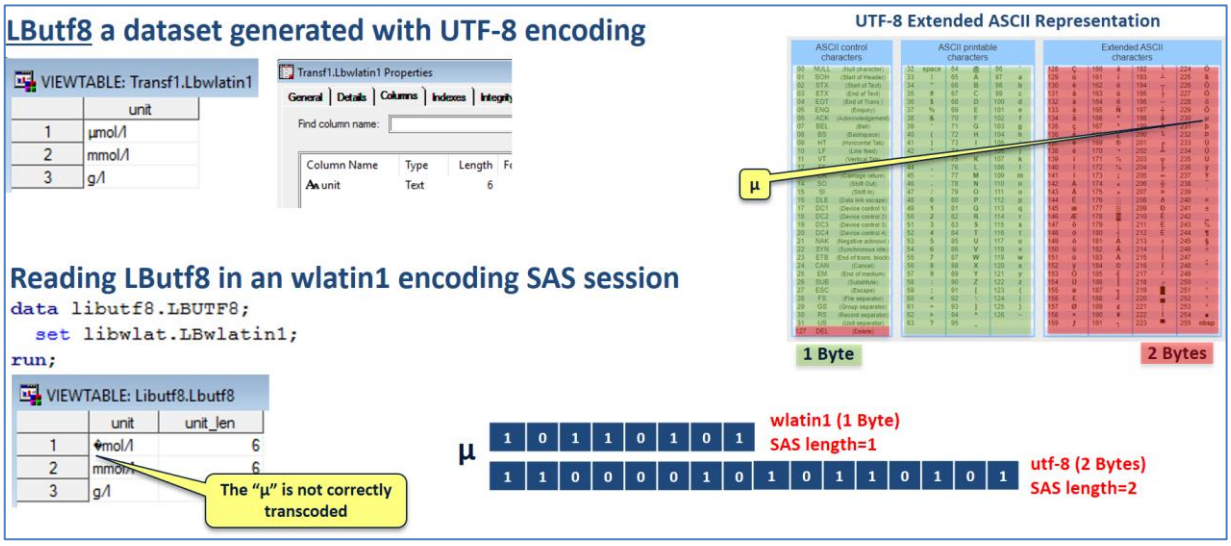

Figure 9: From wlatin1 to UTF-8

One way of fixing this length issue is to use the **CVP** option with the **LIBNAME**. To see the effect, because the CVP option when used makes the SAS library read-only, we need to create the UTF-8 transcoded datasets in a separate

SAS Library. What will be the effect of the CVP option? Well first the micro symbol is now correctly transcoded and interpreted in the newly created dataset, but to make this possible SAS with the CVP option had to increase the length of the UNIT variable from 6 to 9 bytes to correctly represent the micro symbol in the new UTF-8 encoding (figure 10).

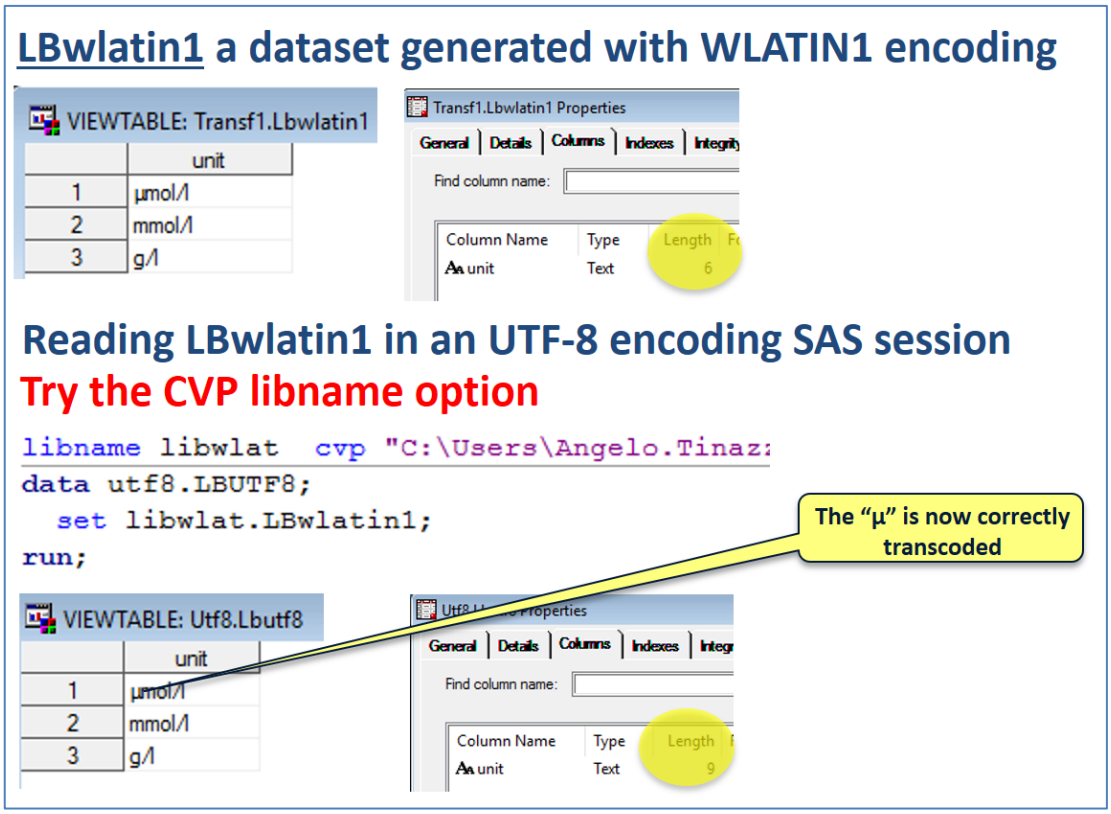

Figure 10: Fixing transcoding issue from wlatin1 to UTF-8 using the LIBNAM CVP option

From UTF-8 to WLATIN1

Let us now consider the same example but by assuming the dataset we received has been created in a UTF-8 encoding environment and now we want to use the dataset in a wlatin1 encoding environment. If we do SET the UTF8 dataset we can see again the micro symbol was correctly transcoded and interpreted in the wlatin1 environment (figure 11).

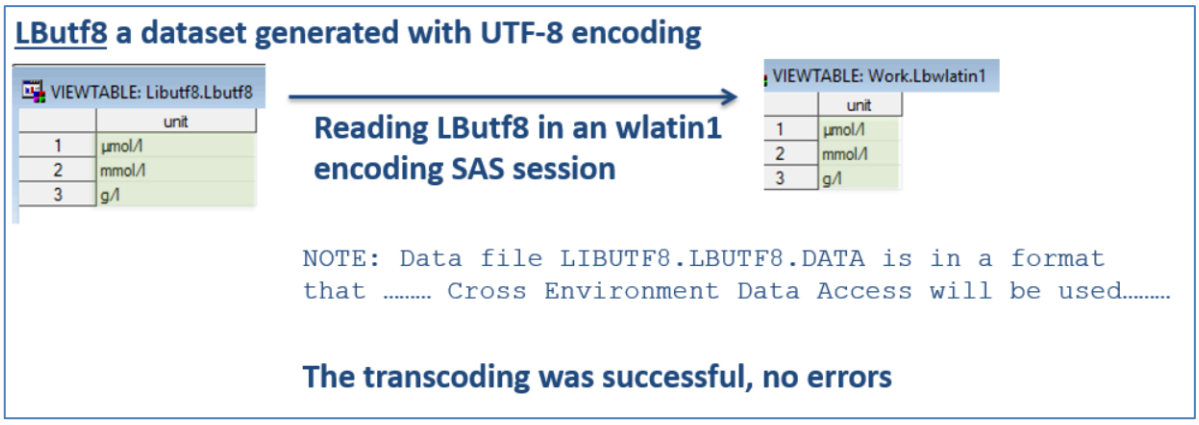

Figure 11: From UTF-8 to wlatin1

Failing transcode to wlatin1

If we look back at the earlier Japanese encoded dataset, if we stay in a wlatin1 encoded environment, we can also notice that if we try to use that dataset, the SAS datastep end with an error (figure 12).

```
data Adveve_1_0;
  set SJISRaw.Adveve_1_0;
run;
ERROR: Some character data was lost during transcoding in the dataset SJISRAW.ADVEVE_1_0. Either the
NOTE: The DATA step has been abnormally terminated.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.ADVEVE_1_0 may be incomplete.  When this step was stopped there were 0 obs
WARNING: Data set WORK.ADVEVE_1_0 was not replaced because this step was stopped.
```

Figure 12: Transcoding causing errors

Many SAS programmers solve this issue by using the **`encoding=ANY`** option which tells SAS to not use the SAS Cross Environmental Data Access and to ignore the current SAS session encoding (figure 13).

```
data Adveve_1_0;
  set SJISRaw.Adveve_1_0(encoding=ANY);
run;
NOTE: There were 212 observations read from the data set SJISRAW.ADVEVE_1_0.
NOTE: The data set WORK.ADVEVE_1_0 has 212 observations and 50 variables.
```

Figure 13: Use of the encoding=ANY option

So the encoding=ANY option might seem the approach with best pros and cons balance as it removes the risk of having errors in the SAS program. However, it must be used carefully because by ignoring the local encoding, SAS anyhow try to make an interpretation the value as something.
So if we try to read in a wlatin1 encoded SAS session the lab dataset in figure 14 originated in a UTF-8 encoding SAS session, SAS will not transcode characters that have a different representation and therefore again the “micro” symbol will be not correctly interpreted.

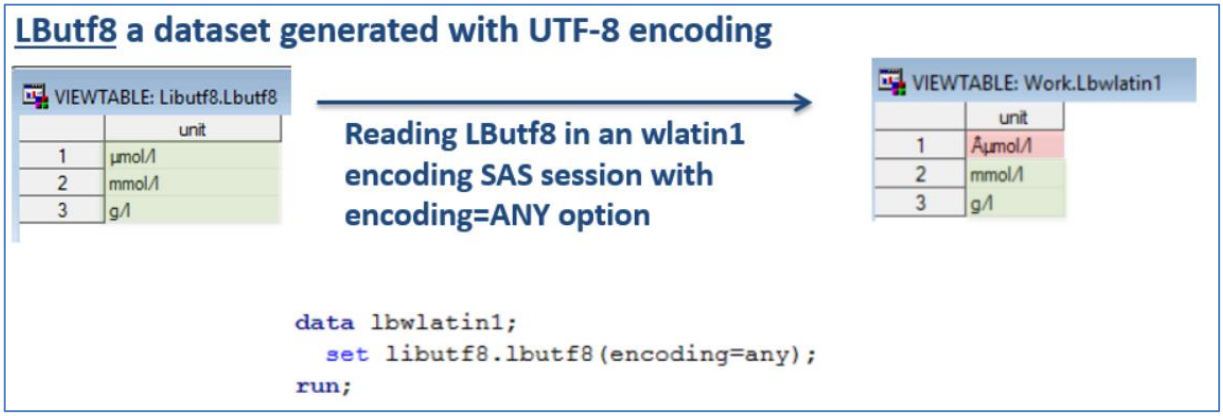

Figure 14: Failing to read UTF-8 dataset in a wlatin1 environment

In such case, we might need to take care of the transcoding and somehow manually handle the transcoding of characters having a different representation, for example by using the **`KCVT`** function as per the example in figure 15.

Transcode datasets to your local encoding

One last recommendation I want to give is that it also makes sense to permanently convert datasets generated in other encoding to your local encoding before using them. This will save some SAS resources and execution time.
In fact, if you have a look at the log when the Cross Environment Data Access enter into action it alerts you it will take some additional CPU resources, as shown on figure 16.
The conversion can be done by simply running a **`PROC COPY`** where all datasets of the source library are copied to the new library using the local session encoding. This can save later some CPU time every time the transcoded datasets are used as shown in figure 16.

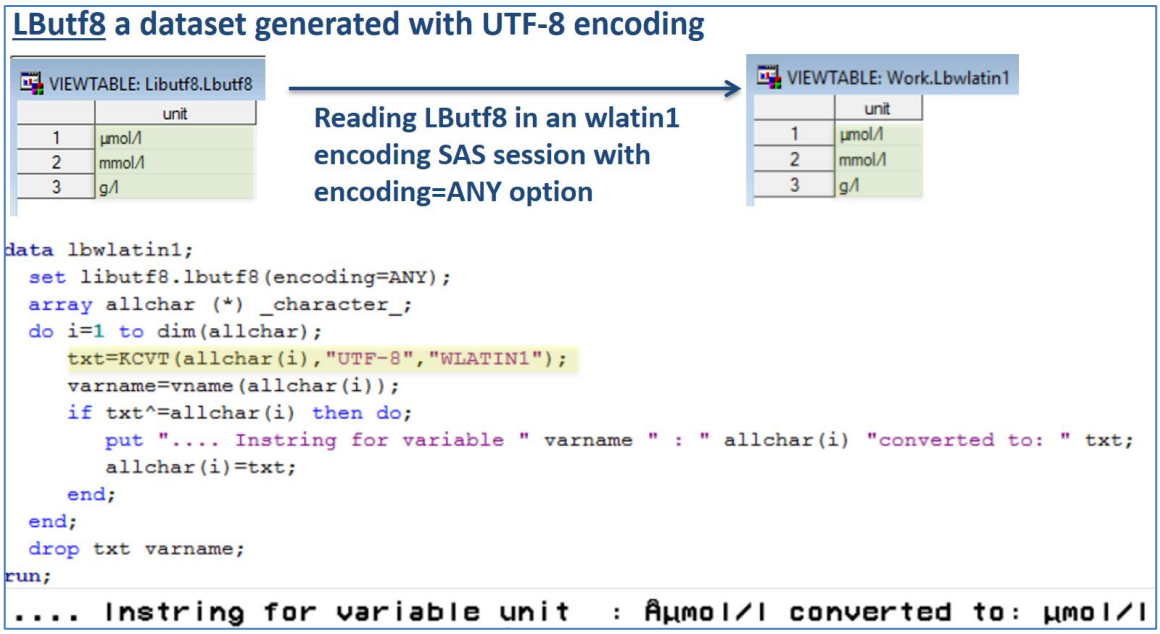

Figure 15: Fixing transcoding issue

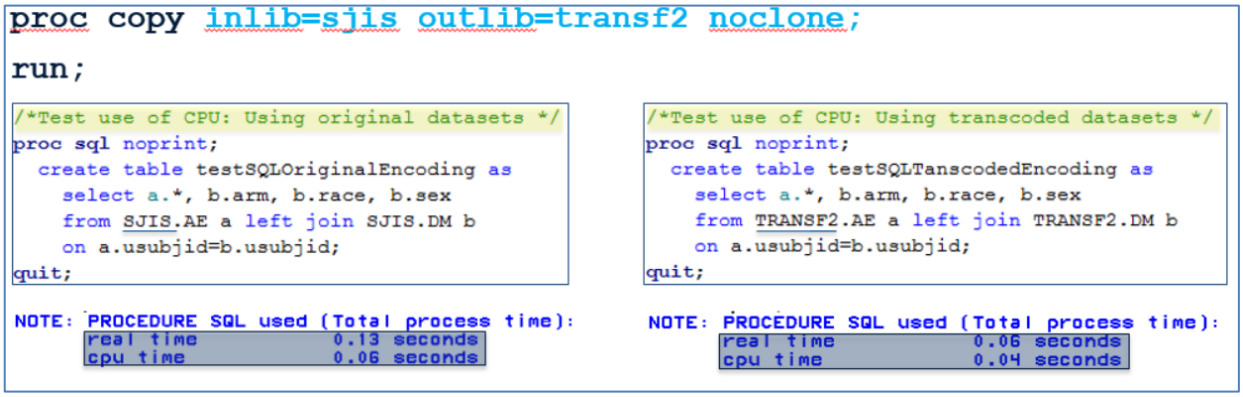

Figure 16: Transcode datasets to your local encoding

ENCODING AND DATA SUBMISSION

I want now to quickly touch the topic of encoding and data submission and more in general the requirements for special characters when datasets are submitted to the Health Authorities such as the FDA.

FDA

If we take for example the FDA "Study Data Technical Conformance Guide," the agency defines some rules with regards to character to be used when naming and labeling datasets and variables but also datasets content where they recommend restricting the use of standard ASCII printable values below 128, so essentially all characters requiring one byte regardless of encoding used as we previously discussed. The FDA also alerts you on possible transcoding issue when working with UTF-8 encoding and they also explicitly ask you to avoid the use of specific range of characters from the ASCII extended set (figure 17).

Regarding the first requirement of avoiding nonprintable characters, or ASCII control characters, one uncomplicated way is the use of the compress function using the compress function modifier to remove all non-printable characters.

```
mychar=compress(mychar,,'kw');
```

For the second requirement, the recommendation is to carefully assess where these characters have been used and find a replacement strategy [6].

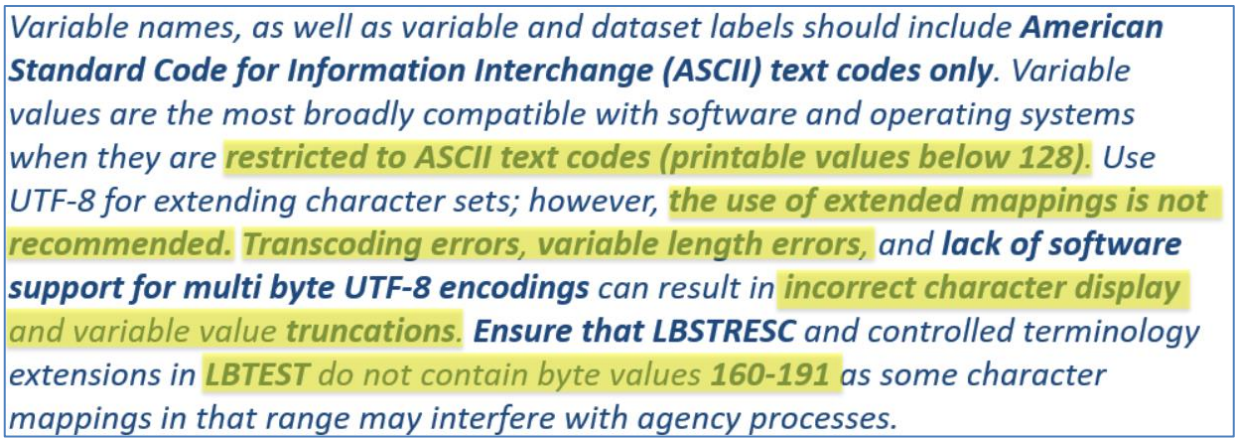

Figure 17: FDA Study Data Technical Conformance Guide section 3.3.5

PMDA AND NMPA

For other agencies, such as the Japanese PMDA or more recently the Chinese NMPA, the tricky part is on the requirement of providing some items in the data submission packages in either Chinese or Japanese; for example AE verbatim in the dataset or even more the define and reviewer guide in Chinese for the NMPA submission. Both PMDA and NMPA, as for the FDA, require the submission of datasets in SAS XPT format.

THE CHALLENGE OF MBCS AND SAS XPT FORMAT

Having a proper setting of the encoding is just a step forward for properly handling Japanese or Chinese characters in SAS. Unfortunately, SAS XPT V5 might pose additional challenges when English text are translated into Chinese, such as altering the length of the required bytes and therefore the length of the variable, length of the variable that might exceed the limit imposed by SAS XPT5.

The translation of some items from English to Chinese in your datasets might expose you to some side effects as demonstrated by Jozef Aerts [9] in his poster with this simple example with the CMTRT and HODECOD label (see example in figure 18).

Concept	Description
Bit and bytes and characters representation	Characters are stored as a series of bytes, where 1 byte = 8 bits e.g. “Character” is intended a letter, a number, a symbol, etc. “Sequence” of bits can represent an integer number i.e. with 1 one byte, numbers between 0 and 255 (2^8-1=256)
Encoding	Encoding is the way a computer interprets and represents the “Characters” within the data by assigning an integer number to each “Character” (code page)
Character Set	Character Set is the set of characters used by a language or a group of language e.g. English or more in general western languages, Chinese, Japanese, etc.
Encoding Method	An Encoding Method is a set of rules that assign the numeric representations to the set of characters
Encoding and SAS	Encoding in SAS establishes the default working environment for your SAS session i.e. WLATIN1 in a Windows Operating System for US and Western European Languages
Transcoding	Transcoding is the process of converting data from one encoding to another. The same character could have different numeric representations in two different encodings
SAS Cross-Environment Data Access (CEDA)	SAS transcoding engine to automatically transcode characters from an encoding to another

Table 2: Summary of key encoding concepts covered in the paper

CONCLUSION

Moving datasets across different environments using different encoding, might cause some data transcoding issue. Therefore, it is important programmers check as a first step when receiving datasets from external sources such as vendors, the encoding used. Hopefully SAS makes available a number of tools and techniques to facilitate the proper conversion, or transcoding, from one encoding to another.

There also consideration related to encoding that I did not cover in this paper, such as dealing with length of variables when encoding is not SBCS and the use of “K” functions [2][3].

Figure 18: Issue with XPT dataset when translating English labels onto Chinese

REFERENCES

1. Tips and Fixes for Cross-Environment Batch Transfer of SAS® Data – Y. Zhuo; PharmaSUG 2018
2. Data Encoding: All Characters for All Countries – D. Dutton; PHUSE 2015
3. The impact of Change from wlatin1 to UTF-8 in SAS Environment – H. Song, A. Koster; PharmaSUG 2016
4. UTF What? A Guide for Handling SAS Transcoding Errors with UTF-8 Encoded Data – M. Stackhouse, L. Pogula, PharmaSUG 2018
5. SAS 9.3 UTF-8 Encoding Support and Related Issue Troubleshooting – J. Liang, Edmonton User Group 2016
6. Non-Printable and Special Characters? … BYTE me! – L. Sims, PHUSE 2016
7. SAS® 9.4 National Language Support (NLS): Reference Guide, Fifth Edition, SAS https://documentation.sas.com/api/collections/pgmsascdc/9.4_3.5/docsets/nlsref/content/nlsref.pdf?locale=en#nameddest=n04275wmuchnqjn1hfx0yzpopw5a
    a. Chapter 3 “Encoding for NLS”
    b. Chapter 4 “Transcoding for NLS”
    c. Chapter 5 “Double-Byte Characters Sets (DBCS)”
8. Guideline on the Submission of Clinical Trial, NMPA Center for Drug Evaluation Data https://www.nmpa.gov.cn/directory/web/nmpa/images/obbSqc7vwdm0ssrU0enKb7dtd29u9a4tbzUrdTyo6jK1NDQo6mhty5wZGY=.pdf
9. Chinese and Asian characters in SAS Transport 5 datasets: Why that is the worst possible choice, J. Aerts, 2020 (unpublished CDISC-US Interchange poster) http://xml4pharma.com/publications/Poster_Jozef_Aerts_Chinese_characters_XPT.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Angelo Tinazzi
Cytel Inc.
Route de Prè-Bois 20
1215 Geneva – Switzerland
Email: angelo.tinazzi@cytel.com
Web: www.cytel.com