

UTF What? A Guide for Handling SAS Transcoding Errors with UTF-8 Encoded Data

Michael Stackhouse¹ and Lavanya Pogula²; Covance Inc., ¹Cary NC; ²Austin, TX

Introduction

The acronyms SBCS, DBCS, or MBCS (i.e. single, double, and multi-byte character sets) mean nothing to most statistical programmers. Many do not concern themselves with the encoding of their data, but what happens when data encoding causes SAS to error? The errors produced by SAS and some common workarounds for the issue may not answer important questions about what the issue was or how it was handled.

Though encoding issues can apply to any set of data, this poster will be geared towards an SDTM submission. Among other topics, in this poster we first explore UTF-8 encoding, explaining what it is and why it is. Furthermore, we demonstrate how to identify the issues not explained by SAS, and recommend best practices dependent on the situation at hand. Lastly, we review some preventative checks that may be added into SAS code to identify downstream impacts early on. By the end of this paper, the audience should have a clear vision of how to proceed when their clinical database is using separate encoding from their native system.

What is encoding?

Encoding is how a computer interprets and represents the values within data. It consists of a character set and uses a coded character set to get from byte to characters.

► LATIN1

Sing Byte Character Set (SBCS)

All characters are stored in a single byte

ASCII character set (basic English characters, 0-9, some special and control characters (e.g. \$, *, or carriage returns)

Extended ASCII (other Western European characters (e.g. ñ, ø, ü, á), along with other common special characters (e.g. ©, ¿, ¡)

► UTF-8

Multi-Byte Character Set (MBCS)

Character may be represented by 1 to 4 bytes

More than 120,000 characters covering 129 scripts and multiple symbol sets

1st 128 characters the same as ASCII

Most extended ASCII characters and beyond require 2 to 4 bytes (e.g. ñ, ø, á, ©, ¿, ¡)

Where You Are and Where You Need to Go

The FDA requires only the English language and expects sponsors to remove or convert non-ASCII characters collected in research before submission. When reading data into a SAS session, the session encoding need not be the same as the datasets' encoding. Check your session encoding:

```
proc options option=encoding; run;
Log Output:
ENCODING=LATIN1 Specifies the default character-set
encoding for the SAS session.
```

Figure 1. How to check your session encoding.

Check your dataset encoding:

```
proc contents data=utf8.myDS; run;
```

From PROC CONTENTS output:

Label	
Data Representation	LINUX_IA64
Encoding	UTF-8 Unicode (UTF-8)

Figure 2. How to check your dataset encoding.

Where Problems Come From in Your Data

In most cases, SAS handles transcoding for you using Cross Environment Data Access (CEDA) (available in SAS 9.2 and later)

```
NOTE: Data file RAWDATA.AE.DATA is in a format that is native to another host, or
the file encoding does not match the session encoding. Cross Environment Data
Access will be used, which might require additional CPU resources and might reduce
performance.
```

Figure 3. CEDA used for transcoding in SAS.

Presented at PhUSE US Connect 2018

But this may not be handled by SAS all the time:

```
ERROR: Some character data was lost during transcoding in the dataset DATA.AE.
Either the data contains characters that are not representable in the new encoding
or truncation occurred during transcoding.
```

Figure 4. Transcoding error.

Transcoding errors can occur for 2 reasons

- The character being transcribed exists in the dataset encoding but not the session encoding.
- A value in the dataset was truncated, resulting in an unreadable character.

How SAS Transcodes and Why Errors Happen

Assuming the character exists in both the dataset encoding and session encoding

- When a SAS length is set on a character variable, the length is not in reference to the number of characters that may be contained, but rather the number of bytes.
 - SCBS: 1 byte = 1 character
 - MBCS: 1 character may consist of 1 to 4 bytes

```
► LATIN1:
01110011:01101111:01100110:1100001
| s | o | f | á |
► UTF-8:
01110011:01101111:01100110:11000011:10100001
| s | o | f | á |
► Truncated UTF-8 Value if length=$4:
01110011:01101111:01100110:11000011
| s | o | f | ? |
```

Figure 5. How SAS transcodes.

The Danger of Encoding = ANY|ASCIIANY

```
data myDS;
set utf8.myDS (encoding=asciiany);
run;
► Truncated UTF-8 Value if length=$4:
01110011:01101111:01100110:11000011
| s | o | f | ? |
► Encoding = ANY|ASCIIANY interprets the values as:
01110011:01101111:01100110:11000011
| s | o | f | Á |
```

Figure 6. Danger of Encoding = ANY|ASCIIANY.

Binary	UTF-8	ASCII
11000011:10100001	á	Ä
11000011:10111100	ü	Å
11000011:10111000	ø	Ä
11000011:10110001	ñ	Ä±

Figure 7. Difference in interpretation for UTF-8 and ASCII encoding.

Identifying and Isolating Your Issues

- Isolating the non-ASCII characters
<variable>=prxchange('s/[x20-x7F]/', -1, <source>);
- Locating issue characters in the data
<variable>=prxchange('s/[^\x20-\x7F]/XX', -1, <source>);
- Removing the “bad” characters
<variable>=prxchange('s/[^\x20-\x7F]/', -1, <source>);

Input:

Obs	bad char
1	ß†ÿöûñ@ØÊµø

Output:

Obs	var1	var2	var3	var4	var5	var6	var7
1	Be†	yoû	c@ñ't	ge†	@id	Øf	mÊ
2	Bµ†	why	w@pld	y@p	want	tø	anyway?
3	Because	s@me	characters	d@n't	like	tø	transc@de

Examples and Tips for Resolving Encoding Issues

The easiest way to eliminate transcoding issues is to match your SAS session encoding to your dataset encoding

- This is not always a possibility
- There are many implications to consider

From this point forward, we will assume that your SAS session encoding has been switched to UTF-8

- Problems with DBCS data
data vitals;
var1="97°F";
var1_translate=translate(var1,"", '');
run;

Obs	var1	var1_translate
1	97°F	97 F

The fix:

```
data vitals;
var1="97°F";
var1_translate=translate(var1,'','');
var1_ktranslate=ktranslate(var1,'','');
run;
```

Obs	var1	var1_translate	var1_ktranslate
1	97°F	97 F	97 F

- K-Functions

Regular SAS Function	K Function
LEFT/RIGHT	KLEFT/KRIGHT
TRIM	KTRIM
STRIP	KSTRIP
LENGTH	KLENGTH
SCAN	KSCAN
SUBSTR	KSUBSTR
UPCASE/LOWCASE/PROPCASE	KUPCASE/KLOWCASE/KPROPCASE
REVERSE	KREVERSE

- CVP Engine

- Use the CVP engine on LIBNAME statements to buffer the lengths of variables to avoid truncation.

Final Thoughts and Recommendations

- Given that issues with non-ASCII characters and transcoding route back to the source, it is best to start these checks at the source itself.
- Review measures for these issues should be put in place throughout the course of a trial to ensure submission readiness
- The easiest way to fix transcoding issues is to have you SAS session encoding match the dataset encoding
- When working in a UTF-8 environment, some special considerations are necessary