

# SAS® GLOBAL FORUM 2021

Paper 1029-2021

## Transcoding: Understand, Troubleshoot, and Resolve a Most Mysterious SAS® Error

Yun (Julie) Zhuo, PRA Health Sciences

### ABSTRACT

In today's integrated and interconnected world, sharing SAS data among organizations and countries has become increasingly common. The transcoding error, which may occur during data transfers between incompatible SAS encoding environments, remains one of the most mysterious SAS errors to programmers. This presentation discusses three steps to demystify the transcoding issues. First, encoding, transcoding and other background information will be explained. Second, SAS procedures, options and other techniques that are useful for troubleshooting will be given. Finally, the character variable padding (CVP) engine and SAS configuration files are provided as solutions to the problem. By the end of the presentation, the audience should have a clear vision for troubleshooting and solving transcoding errors.

### INTRODUCTION

If we live in a perfectly standardized world, sharing data would be a smooth copy-and-paste from computer to computer. Unfortunately, we often receive data from an incompatible computing environment with an incompatible session encoding. Problems thus arise.

SAS Cross-Environment Data Access (CEDA) is designed to tackle the problem of incompatibility encodings. However, it has limitations. For example, data loss or truncation may occur when the data contain extended ASCII or other special characters. When this happens, CEDA will issue a message such as the following:

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

What can we do when confronted with such a mysterious error message? Pushing back the data to the data provider is not always a feasible option. In this paper, we aim to demystify the error message, and provide tools, workarounds, and solutions to help resolve the error at the data recipient's end.

### UNDERSTAND THE TRANSCODING ERROR

Before we attempt to troubleshoot and resolve the transcoding error, it is helpful to understand why the error occurs. In this section, we are going to provide background information to help demystify the error message.

#### WHAT IS ENCODING

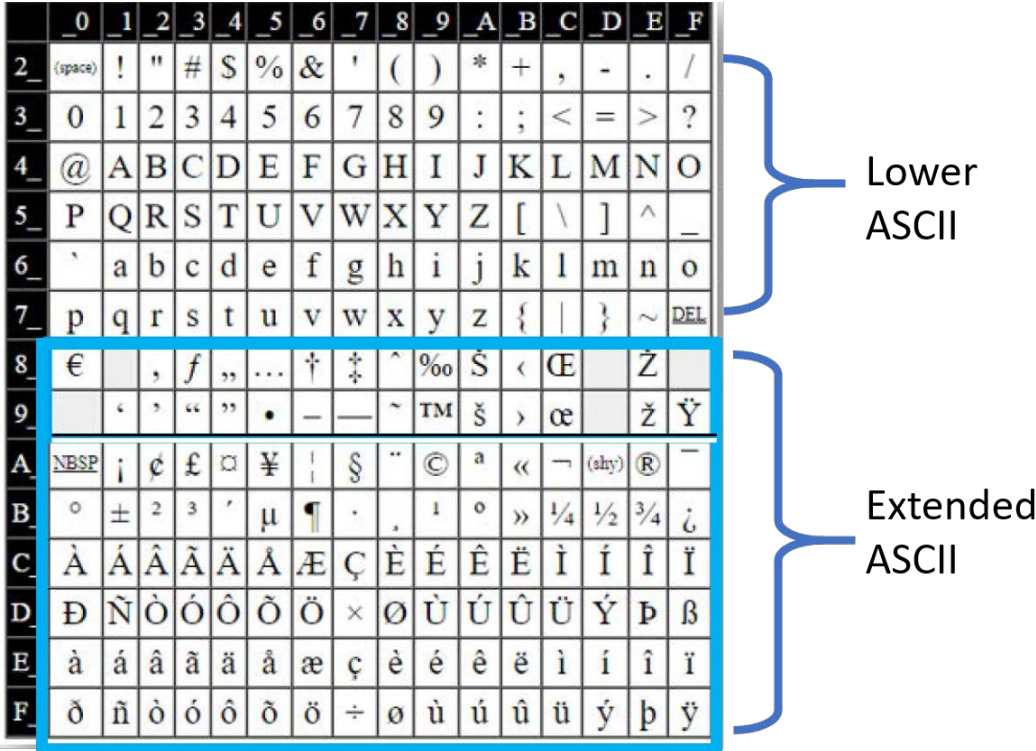
In computers, character data is stored as a series of bytes, which is made up of binary digits called bits. A coded **character set** associates each character with a number. Computer **encoding** maps the bits and bytes of stored data to the correct characters via the coded character set. Since our computing environments are not perfectly standardized, there are different types of encodings in use, and each encoding has its own character set.

Name of Encoding	Associated Character Set	Distinguishing Factors
WLATIN1	ASCII and Extended ASCII character set	Represent 256 characters. Each character is stored in a single byte. Single Byte Character Set (SBCS)
SHIFT-JIS	ASCII, Katakana, and other Japanese characters	Characters are stored in either 1 or 2 bytes. Double Byte Character Set (DBCS)
UTF-8	Unicode character set including ASCII, foreign languages, special symbols, and more	Represent over 120,000 characters. Each character is stored in 1 to 4 bytes. Multi-Byte Character Set (MBCS)

**Table 1. Examples of Commonly Used Encodings**

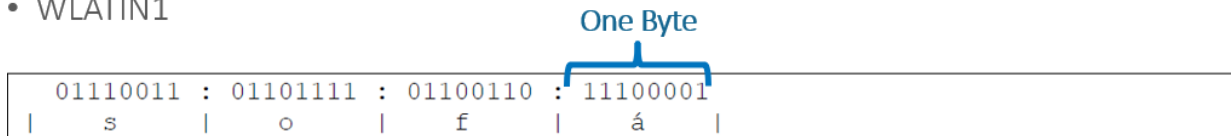
Table 1 gives examples of some commonly used encodings and their associated character sets. The last column of the table shows two distinguishing factors of different data encoding types:

1. The number of characters each encoding can represent differs. While the WLATIN1 encoding, the most widely used encoding type in the Western countries, can only represent 256 lower ASCII, ASCII control, and extended ASCII characters, the UTF-8 encoding, a common encoding form of the Unicode standard, can represent over 120,000 characters. Figure 1 shows all the lower ASCII and extended/upper ASCII characters.
2. The storage size of each encoding differs. While the single byte character set (SBCS) only takes up one single byte for each character, the multiple byte character set (MBCS) needs up to four bytes to store characters. For example, Figure 2 shows that in order to represent the extended ASCII character 'á', the WLATIN1 encoding needs only one byte, but the UTF-8 encoding uses two bytes to represent the character.

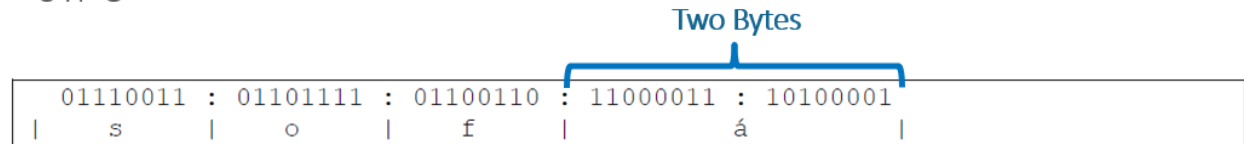


**Figure 1. ASCII and Extended ASCII Characters**

- WLATIN1



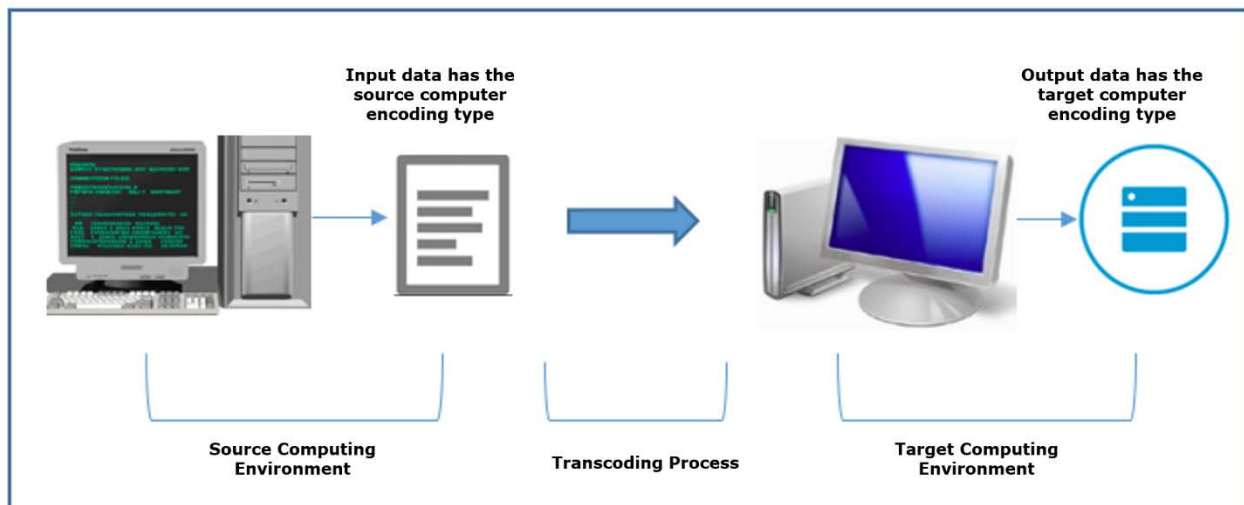
- UTF-8



**Figure 2. Difference in Storage Size for WLATIN1 and UTF-8 Encodings**

### WHAT IS TRANSCODING

Now what is transcoding? Transcoding is the process of converting from one computer encoding to another. In SAS, each programming session has a pre-configured encoding. If the encoding of the input data differs from the encoding of our currently executing SAS session, SAS will invoke the Cross-Environment Data Access (CEDA) engine to transcode the data, automatically converting the input data encoding to the encoding of the SAS session in the target computer. Figure 3 demonstrates the process of transcoding from one computing environment to another.



**Figure 3. Cross-Environment Computer Transcoding Process**

When SAS invokes CEDA and the transcoding occurs, SAS issues the following informational note to the log:

NOTE: Data file 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.

### HOW TRANSCODING ERROR OCCURS

Unfortunately, the SAS CEDA does not work as miraculously as we hope. Transcoding error occurs when the CEDA transcoding process fails. The transcoding error contains two components, as shown below in Figure 4, each pointing to a potential problem that causes the error.

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

**Figure 4. Two Components of the Transcoding Error Message**

The first problem is characters that are not representable in the new encoding. As we explained earlier, some encodings such as the UTF-8 encoding store more than 120,000 characters covering 129 scripts and multiple symbol sets. ASCII characters, Asian languages and non-ASCII special characters are all represented in the UTF-8 encoding. By contrast, the WLATIN1 encoding only contains 256 ASCII and extended ASCII characters. Any character that is not included in the WLATIN1 encoding will be lost if we try to transcode from UTF-8 to WLATIN1. In short, the unrepresentable characters error occurs when the input data set contains characters that do not exist in the target computer encoding, triggering the transcoding error message in the log.

The second problem is character data truncation. As we explained earlier, one distinguishing factor between encodings is the storage size. The single byte character set (SBCS) only takes up one single byte for each character, but the multiple byte character set (MBCS) needs more than one bytes to store characters. Therefore, if the target data set uses a MBCS such as the UTF-8, the length of the character variables defined in the source or input data set may not be big enough to hold all the characters for storage in the target encoding environment. When this occurs, SAS will trigger the transcoding error message to warn you about truncation.

## TROUBLESHOOT THE TRANSCODING ERROR

Now that we understand how transcoding error occurs, the next step is to troubleshoot the issues. There are a number of SAS options and procedures that are helpful. In this section, we will provide answers to specific questions encountered during troubleshooting.

The first question we ask is which of the two components in the error message has caused the problem. Is it unrepresentable characters or data truncation? To answer this question, we need to find out the encodings used in both the source data and the currently executing SAS session. If SAS transcodes SBCS (e.g., WLATIN1) to MBCS (e.g., UTF-8), then the problem is data truncation. On the other hand, if SAS transcodes a large character set to a smaller character set, then the problem is most likely the unrepresentable characters.

### WHAT IS THE ENCODING OF YOUR INPUT DATA

We can find out the encoding of the input data using one of the following two methods.

The first method is the CONTENTS procedure. That is, we run the following simple SAS CONTENTS procedure:

```
proc contents data=source.DM;
run;
```

Then we look for the encoding row in the output:

The CONTENTS Procedure			
Data Set Name	SOURCE.DM	Observations	437
Member Type	DATA	Variables	27
Engine	V9	Indexes	0
Created	02/01/2021 08:40:34	Observation Length	360
Last Modified	02/01/2021 08:40:34	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label	Demographics		
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

**Figure 5. The Encoding Attribute in the PROC CONTENTS Output**

The example in Figure 5 shows that the input data set SOURCE.DM was created under the Windows computing environment using the WLATIN1 encoding. If the transcoding error occurs and the currently executing SAS session uses a different encoding with higher byte character set (e.g. UTF-8), then the problem is data truncation.

If you see the following data representation and encoding values instead in the PROC CONTENTS output:

```
Data Representation HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64
Encoding utf-8 Unicode (UTF-8)
```

then the input data set was created under the UTF-8 encoding. If the transcoding error occurs and the currently executing SAS session uses a different encoding with a smaller character set (e.g. WLATIN1), then the problem is unrepresentable characters.

The alternative method to find out the encoding of the input data is the ATTRC function.

After we run the following SAS codes:

```
%let dsn=source.DM;
%let dsid=%sysfunc(open(&dsn,i));
%put &dsn ENCODING is: %sysfunc(attrc(&dsid,encoding));
%let rc=%sysfunc(close(&dsid));
```

The encoding will be printed to the SAS log as shown below:

```
34          %put &dsn ENCODING is: %sysfunc(attrc(&dsid,encoding));
source.DM ENCODING is: wlatin1 Western (Windows)
```

## WHAT IS THE ENCODING OF OUR CURRENT SAS SESSION

We can find out the encoding of our currently executing SAS session using one of the following two methods.

The first method is the OPTIONS procedure. That is, we run the following simple SAS OPTIONS procedure in our current SAS session:

```
proc options option=encoding;
run;
```

Then we look for the encoding information printed to the SAS log.

```
32      proc options option=encoding;
33      run;

SAS (r) Proprietary Software Release 9.4 TS1M3

ENCODING=WLATIN1 Specifies the default character-set encoding for the SAS session.
NOTE: PROCEDURE OPTIONS used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

**Figure 6. The Encoding Attribute in the SAS Log from the OPTIONS Procedure**

The example in Figure 6 shows that the current SAS session in our computer uses the WLATIN1 encoding to represent character data.

Alternatively, we can use the GETOPTION function. That is, we run the following SAS code in our current SAS session:

```
%put encoding=%sysfunc(getoption(encoding));
```

We will find the similar encoding information printed to the SAS log. As shown below:

```
31
32      %put encoding=%sysfunc(getoption(encoding));
encoding=WLATIN1
33
```

The above examples show that the current SAS session uses WLATIN1, the encoding with single byte character set. If transcoding error occurs and the input data has a larger character set, then the problem is the unrepresentable characters.

## WHERE ARE THE UNREPRESENTABLE CHARACTERS

Unrepresentable characters are often buried under mountains of data on hand. A single unrepresentable character in an inconvenient location can stop SAS from transcoding. It is helpful if we can identify the unrepresentable characters causing transcoding issues and visually inspect them before we make decisions on our next move.

In the example below, shown in Figure 7, input data set has the UTF-8 encoding but the current SAS session in our computer uses the WLATIN1 encoding. We use a simple data step to read the input data set. Not surprisingly, a transcoding error occurs. In the log, in addition to the transcoding error message, SAS also prints a warning message indicating that the data step stopped right after it finished processing 799 observation. What it means is that on observation number 800, there must be some unrepresentable characters that stopped the data step from processing.

```

7  data indata.cm4;
8  set rawdata.cm4;
INFO: Data file RAWDATA.CM4.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.
9  run;

ERROR: Some character data was lost during transcoding in the dataset RAWDATA.CM4. Either the data
contains characters that are not representable in the new encoding or truncation occurred
during transcoding.
NOTE: The DATA step has been abnormally terminated.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 799 observations read from the data set RAWDATA.CM4.
WARNING: The data set INDATA.CM4 may be incomplete. When this step was stoppd there were 799
observations and 66 variables.
WARNING: Data set INDATA.CM4 was not replaced because this step was stopped.
NOTE: DATA statement used (Total process time):
    real time           1.18 seconds
    user cpu time       0.23 seconds
    system cpu time     0.06 seconds
    memory              1393.06k
    OS Memory          15832.00k
    Timestamp           03/27/2018 12:02:25 PM
    Step Count          3  Switch Count  0

```

**Figure 7. Warning Message from Data Step Shows Where Error Occurs in Data**

To confirm our theory, we use the ENCODING=ASCIANY SAS option to bypass the transcoding error in order to take a visual inspection of the row number 800 in the input data set. Note that the ENCODING=ASCIANY option will allow SAS to ignore the UTF-8 encoding altogether. SAS will interpret every bite in the input data using its single byte ASCII coded values.

The syntax is the simple SAS codes shown below:

```

Data indata.cm4;
  Set rawdata.cm4 (encoding=asciiany);
Run;

```

After we execute the above codes, we will be able to read the entire input data set with 1163 observations without any error or warning messages, as shown below in Figure 8.

```

16  data indata.cm4;
17  set rawdata.cm4(encoding=asciiany);
18  run;

NOTE: There were 1163 observations read from the data set RAWDATA.CM4
NOTE: The data set INDATA.CM4 has 1163 observations and 66 variables.
NOTE: DATA statement used (Total process time):
    real time           0.84 seconds
    user cpu time       0.03 seconds
    system cpu time     0.15 seconds
    memory              1032.25k
    OS Memory          15832.00k
    Timestamp           03/27/2018 12:06:41 PM
    Step Count          6  Switch Count  0

```

**Figure 8. Bypass Transcoding Error with the SAS Encoding=ASCIANY Option**

Then, to locate the unrepresentable characters, we open the output data set we have created without error, scroll down to row number 800, and then as shown in Figure 9, we immediately spot some weird characters that do not make any sense.

	Reported Name of Drug Med or Therapy	Reported Name of Drug Med or TherapyATC
795	REVATIO	ANTIHYPERTENSIVES
796	CALONAL	NERVOUS SYSTEM
797	RIKAVARIN	STOMATOLOGICAL PREPARATIONS
798	XYLOCAINE(LIDOCAINE)	ANESTHETICS FOR TOPICAL USE
799	CARBOCYSTEINE	COUGH AND COLD PREPARATIONS
800	MEDICON <del>æ&amp;</del> 1% DEXTROMETHORPH HYDROBROMIDE HYDRATE1% <del>æ&amp;</del>	OPIUM ALKALOIDS AND DERIVATIVES
801	TALION	OTHER ANTIHISTAMINES FOR SYSTEMIC USE
802	KEFRAL	OTHER BETA-LACTAM ANTIBACTERIALS
803	TOCOPHEROL NICOTINATE	CARDIOVASCULAR SYSTEM

**Figure 9. Locate the Unrepresentable Characters in the Output Data Set**

What happened is that, with the ENCODING=ASCIIANY bypassing option, SAS will interpret every bit in the input data using its single byte ASCII coded values. It is most likely that the unrepresentable characters will be turned into ASCII special characters such as those in the blue rectangle in Figure 1. ASCII and Extended ASCII Characters.

Therefore, the ENCODING=ASCIIANY SAS option provides a way for us to locate the unrepresentable characters, and visually inspect the data. It also helps us estimate the extent of our issues. However, it does not completely resolve our problem.

## SOLUTIONS

The transcoding error has two components. Each component points to a potential reason for the transcoding failure: data truncation or unrepresentable characters. Each component also has its own solutions. If the error is caused by data truncation, we can use the Character Variable Padding (CVP) engine to avoid truncation. If the error is caused by unrepresentable characters, we propose updating the session encoding in the SAS configuration file if easy walk-around solutions do not work for you.

### AVOID DATA TRUNCATION WITH THE CVP ENGINE

If you find out that the encoding of your session uses the MBCS but the encoding of the source data set uses the SBCS or the DBCS, the error message you encountered above usually means that there is not enough space in one or more character columns to convert the data to the target encoding. If this is the case, you should consider using the Character Variable Padding (CVP) engine to avoid truncation.

What is the CVP engine? It is a read-only engine that expands the length of the character variables at your request. It serves as an intermediate engine that is used to prepare the data for transcoding. By default, it multiplies the character variable lengths in the input data set by 1.5. After the lengths are increased, the primary engine is used to do the actual processing.

In the following example, the input data set was created under the Windows computing environment using the WLATIN encoding which represents a SBCS character set. As shown in Figure 10, the input data set contains one character variable and two records. Both



records contain four ASCII and extended ASCII characters. The length of the character variable is set to 4, which provides enough space in the WLATIN encoding environment to store the data because SBCS only takes up one single byte for each character.

	CharVar
1	brød
2	über

Column Name	Type	Length
CharVar	Text	4

**Figure 10. Example Input Data Set with Extended ASCII Characters**

When we try to read the above input data set into a different encoding session such as the UTF-8 encoding SAS session. The character variable will be truncated. Because in the UTF-8 encoding, it takes more than one byte to store special characters such as the 'Ø' and 'Ü' in this example.

To resolve this issue, we run the following code to invoke the CVP engine with a default multiplier of 1.5:

```
libname source cvp 'Source-data-library';
```

After running the above code, the length of the variable will be automatically increased from 4 to 6, as demonstrated below in Figure 11.

Before CVP Engine Processing	After CVP Engine Processing
------------------------------	-----------------------------

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	CharVar	Char	4

➔

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	CharVar	Char	6

**Figure 11. Variable Length Before and After CVP Engine Processing**

In order to multiply the character variable lengths by a different amount, you need to use the CVPMULTIPLIER= option to specify a number. You can specify a value from 1 to 5, or you can specify a value of 0 to let the CVP engine determine the amount. Here is the example code that uses a user-specified multiplier:

```
libname source cvp 'Source-data-library' cvpmultiplier=2.0;
```

It may take some experimenting to determine the correct multiplier in order to make sure you have adequate space for all situations. During the experimenting process, be sure to review the log carefully and watch for truncation error messages.

Since the CVP engine is read-only, an additional LIBNAME statement is required in order to save a permanent copy of the converted data set. Here is the full example code:

```
libname source cvp 'Source-data-library';
libname target 'Target-data-library';
proc copy noclone in=source out=target;
run;
```

In the above example, the NOCLONE option in PROC COPY is required. The option makes sure SAS adopt the target operating system data representation, the target session encoding, and other relevant attributes.

To conclude, we run the risk of truncating character data when transcoding into a character set that stores data in multiple bytes. By expanding the length of the character variables,

the CVP read-only LIBNAME engine is an effective and elegant solution to avoid data truncation.

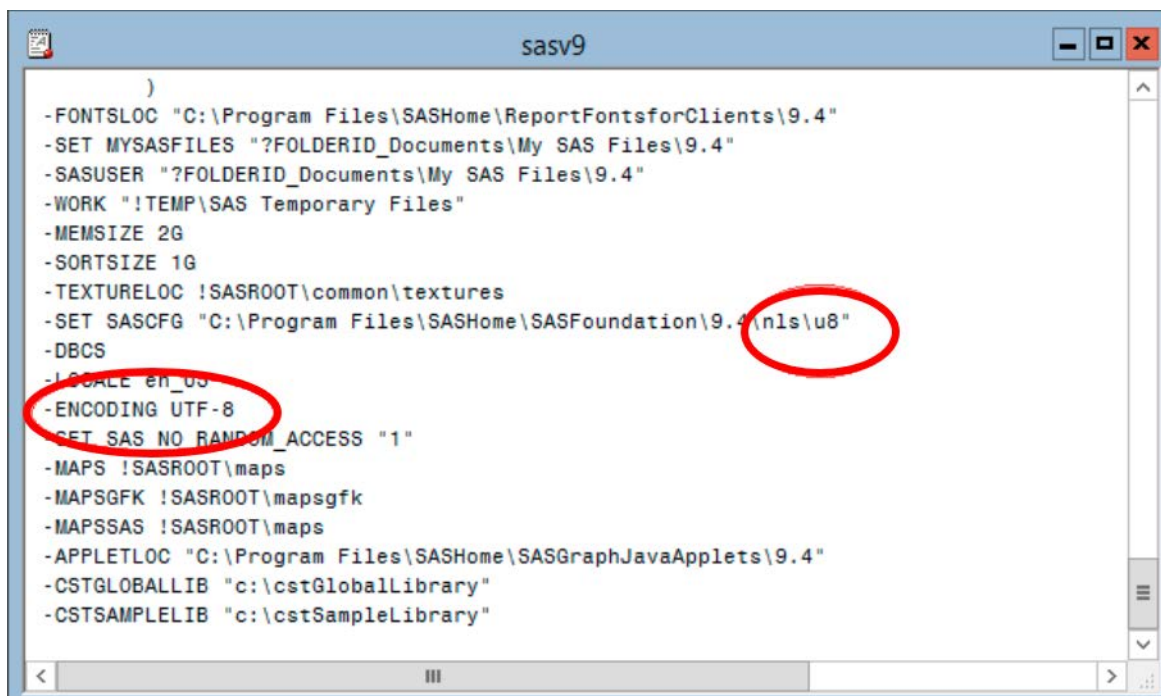
## UPDATE SAS SESSION ENCODING FOR UNREPRESENTABLE CHARACTERS

If you find out that the source data encoding is UTF-8 but our SAS session encoding is WLATIN1 or any encoding other than the comprehensive UTF encoding, the problem is most likely the unrepresentable characters because some characters in the UTF-8 character set do not exist in other encoding types. Although the CVP engine resolves the issue of truncation, it is not helpful if the transcoding error was triggered by the characters that are not representable in the target encoding in our SAS session.

What can we do? The first impulse is to contact the data provider and ask them to remove or replace all the unrepresentable characters. However, this will involve negotiations with outside organizations, and it may not be feasible or convenient in many instances.

The solution is to update the encoding type in our currently executing SAS session. We can update the encoding type to either the UTF-8 encoding or the same encoding of the source data so that all characters in the source data can be represented in the SAS session in our computer.

The encoding type of our SAS session is configured in the SAS configuration file. Figure 12 below is a partial snapshot of an example SAS configuration file that defines the encoding of the SAS session as UTF-8.



```
)
-FONTSLOC "C:\Program Files\SASHome\ReportFontsforClients\9.4"
-SET MYSASFILES "?FOLDERID_Documents\My SAS Files\9.4"
-SASUSER "?FOLDERID_Documents\My SAS Files\9.4"
-WORK "!TEMP\SAS Temporary Files"
-MEMSIZE 2G
-SORTSIZE 1G
-TEXTURELOC !SASROOT\common\textures
-SET SASCFG "C:\Program Files\SASHome\SASFoundation\9.4\nls\us8"
-DBCS
-SCALE en_us
-ENCODING UTF-8
-SET SAS NO_RANDOM_ACCESS "1"
-MAPS !SASROOT\maps
-MAPSGFK !SASROOT\mapsgfk
-MAPSSAS !SASROOT\maps
-APPLETLOC "C:\Program Files\SASHome\SASGraphJavaApplets\9.4"
-CSTGLOBALLIB "c:\cstGlobalLibrary"
-CSTSAMPLELIB "c:\cstSampleLibrary"
```

**Figure 12. Encoding Definition in the SAS Configuration File**

To update the SAS session encoding, firstly, we need to locate the SAS configuration file in our computer. After running the following OPTIONS procedure:

```
Proc options option=config;
Run;
```

The location of the SAS configuration file will be printed to the log, as show below in Figure 13.

```

1  proc options option=config;
2  run;

SAS (r) Proprietary Software Release 9.4 TSIM2

CONFIG=C:\Program Files\SASHome\SASFoundation\9.4\nls\en\SASV9.CFG
Specifies the configuration file that is used when initializing or overriding
the values of SAS system options.

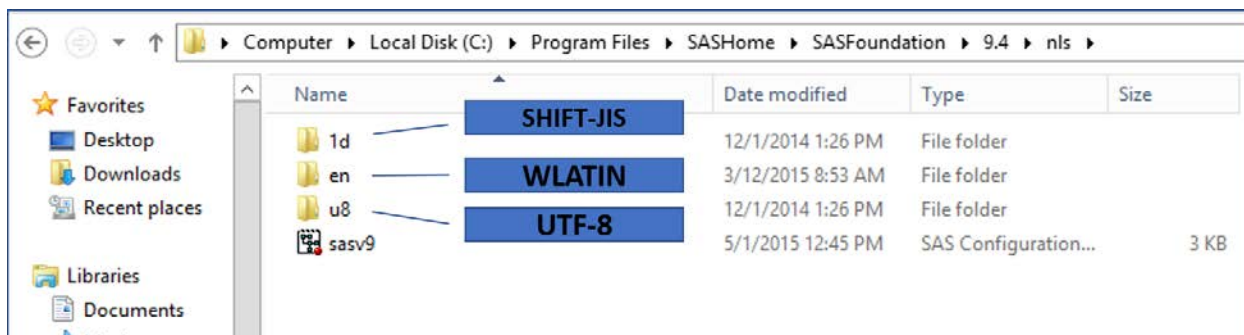
NOTE: PROCEDURE OPTIONS used (Total process time):
real time          0.06 seconds
user cpu time      0.00 seconds
system cpu time    0.00 seconds
memory            19.09k
OS Memory         8672.00k
Timestamp         03/26/2018 01:30:49 PM
Step Count        2  Switch Count  0

```

**Figure 13. Locate the SAS Configuration File using the OPTIONS Procedure**

By default, during SAS installation, SAS installs three configuration files to our computer. Figure 14 shows three default sub-folders and each sub-folder contains a copy of the SAS configuration file with a specific type of encoding.

Subfolder '1d' contains a copy of the configuration file with the encoding type of SHIFT-JIS. Subfolder 'en' contains a copy of the configuration file with the encoding type of WLATIN1. Subfolder 'u8' contains a copy of the configuration file with the encoding type of UTF-8. More encoding types could be installed through a customized SAS installation.



**Figure 14. Default locations for SAS Configuration Files**

The second step is to instruct SAS to use a copy of the SAS configuration file that has the desired type of encoding: the UTF-8 encoding or the encoding used in the data.

If you use the Windows platform, you can customize your shortcut icon to instruct SAS to invoke UTF-8 by following the steps below, as illustrated in Figure 15.

1. Locate your SAS shortcut icon or your SAS Enterprise Guide shortcut icon. If you do not have a SAS shortcut icon already created, you have the option of creating one before customizing it following the below steps.
2. Right-click the shortcut icon and select **Properties**.
3. On the **Shortcut** tab, in the **Target** line, update to instruct SAS to use the UTF-8 version of the NLS (National Language Support) configuration file (sasv9.cfg). For example:

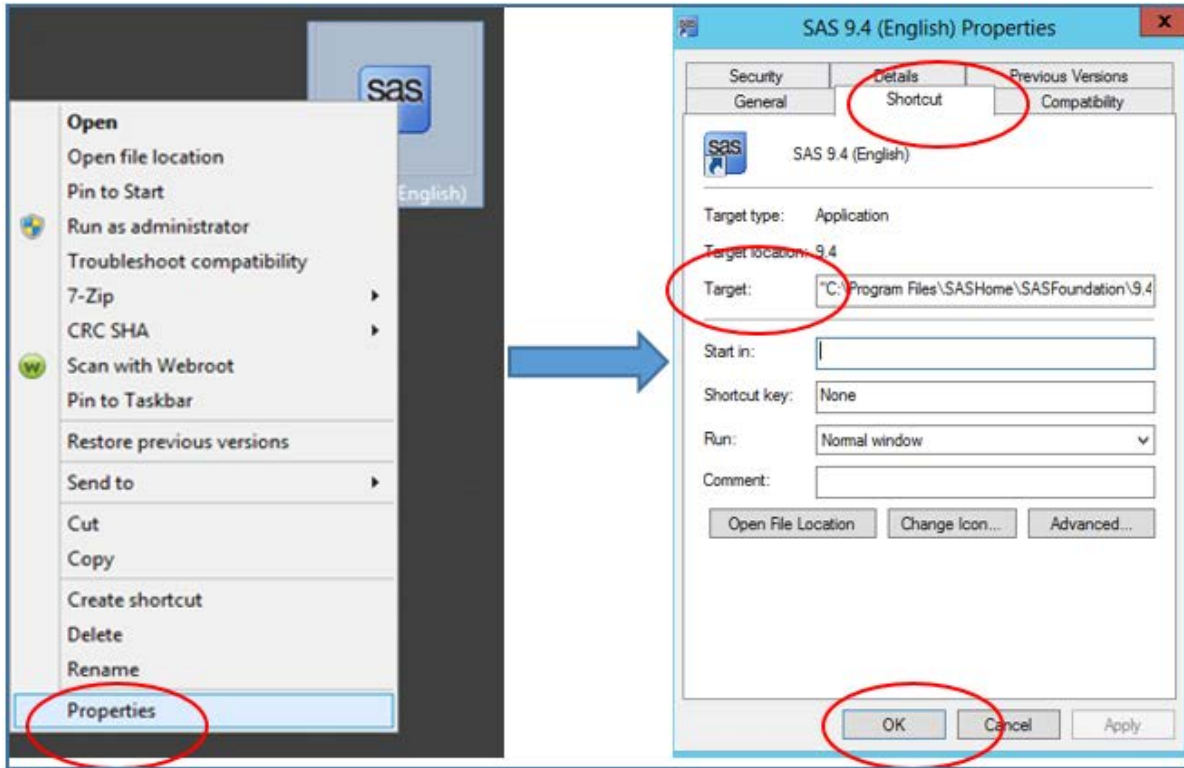
```

"C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -CONFIG
"C:\Program Files\SASHome\SASFoundation\9.4\nls\u8\SASV9.CFG"

```

4. Click OK.

- Optionally, you can rename the shortcut icon to indicate this shortcut will invoke SAS with the UTF-8 encoding. On your desktop, you can also create additional shortcut icons that have other types of encoding.



**Figure 15. Update SAS Session Encoding through the SAS Shortcut Icon**

If you invoke SAS through a command line or a batch file, you have the option to specify the configuration file location through the CONFIG system option. For example, the code below invokes SAS using the UTF-8 encoding version of the configuration file:

```
C:\Progra~1\SASHome\SASFou~1\9.4\sas.exe
-sysin ProgramName.sas
-config C:\Progra~1\SASHome\SASFou~1\9.4\nls\u8\SASV9.CFG
```

For detailed instruction on invoking SAS with different language and encoding sessions on the Windows platform, refer to the SAS Technical Paper: Setting up SAS9 National Language Support in Microsoft Windows Operating Environments.

If you use the UNIX platform, SAS is invoked by Bourne Shell scripts. The invocation scripts are named using the language codes of the installed language. For example, sas\_en invokes the English version, and sas\_u8 invokes the Unicode version. Follow the steps below to update your session encoding:

- Locate the invocation script under the following directory:  
!SASROOT/bin
- Update the script to change the SAS session locale or encoding.

For detailed instruction on invoking SAS with different encoding sessions on the UNIX platform, refer to SAS Configuration Guide for SAS 9.4 Foundation for UNIX Environments.

To conclude, the encoding of our SAS session may not be able to accommodate special characters in the source data set. After we convert our SAS session encoding to either UTF-8 or a compatible type of encoding, we will be able to avoid the transcoding error. We will be able to work with all special characters in the data including foreign languages and other non-ASCII characters.

## CONCLUSION

We may encounter the mysterious transcoding error when we try to work with data from a different encoding environment. This paper shows that if we understand computer encoding and how the transcoding occurs, the problem is not as mysterious as it appears to be.

There are two potential reasons for the transcoding failure. If the reason is data truncation, we can use the CVP engine to expand the length of the character variables in the data. On the other hand, if the reason is unrepresentable characters, we may have to convert our own SAS session encoding to be compatible with the encoding in the data. Either way, data recipients will be able to resolve the problem at their end.

## REFERENCES

Bouedo, M. 2020. "The SAS Encoding Journey: A Byte at a Time." Proceedings of the SAS Global Forum 2020 Conference, Cary, NC: SAS Institute Inc. Available at <https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4561-2020.pdf>

Carlton, J. 2017. "SAS Blogs: Demystifying and Resolving Common Transcoding Problems." Accessed March 6, 2018. <https://blogs.sas.com/content/sgf/2017/05/19/demystifying-and-resolving-common-transcoding-problems/>

Dutton, D. 2015. "Data Encoding: All Characters for All Countries." Proceedings of the PharmaSUG 2015 Conference, Cary, NC: SAS Institute Inc. Available at <https://www.lexjansen.com/phuse/2015/dh/DH03.pdf>

Li, H. 2020. "Turn Yourself into a SAS® Internationalization Detective in One Day: A Macro for Analyzing SAS Data of Any Encoding." Proceedings of the SAS Global Forum 2020 Conference, Cary, NC: SAS Institute Inc. Available at <https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4645-2020.pdf>

SAS Institute Inc. 2014. Usage Note 52716. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/kb/52/716.html>

SAS Institute Inc. 2015. Setting up SAS® 9 National Language Support in Microsoft Windows Operating Environments. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/techsup/technote/ts801.pdf>

SAS Institute Inc. 2016. Moving and Accessing SAS® 9.4 Files, Third Edition. Cary, NC: SAS Institute Inc. Available at <http://documentation.sas.com/?docsetId=movefile&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en>

SAS Institute Inc. 2017. Configuration Guide for SAS® 9.4 Foundation for UNIX Environments. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/installcenter/en/ikfdtnunxcg/66380/PDF/default/config.pdf#page=33>

SAS Institute Inc. 2017. Migrating Data to UTF-8 for SAS® Viya™ 3.3. Cary, NC: SAS Institute Inc. Available at <http://documentation.sas.com/?cdcId=vdmmIcdc&cdcVersion=8.1&docsetId=viyadatamig&docsetTarget=titlepage.htm&locale=en>

SAS Institute Inc. SAS Technical Paper: Multilingual Computing with SAS® 9.4. Cary, NC: SAS Institute Inc. Available at [https://support.sas.com/resources/papers/Multilingual\\_Computing\\_with\\_SAS\\_94.pdf](https://support.sas.com/resources/papers/Multilingual_Computing_with_SAS_94.pdf)

Stackhouse, M. 2018. "UTF What? A Guide for Handling SAS Transcoding Errors with UTF-8 Encoded Data." Proceedings of the PharmaSUG 2018 Conference, Cary, NC: SAS Institute Inc. Available at <https://www.pharmasug.org/proceedings/2018/BB/PharmaSUG-2018-BB08.pdf>

Zhuo, Y. 2018. "Tips and Fixes for Cross-Environment Batch Transfer of SAS® Data." Proceedings of the PharmaSUG 2018 Conference, Cary, NC: SAS Institute Inc. Available at <https://www.pharmasug.org/proceedings/2018/BB/PharmaSUG-2018-BB14.pdf>

## CONTACT INFORMATION

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

Yun (Julie) Zhuo  
PRA Health Sciences  
ZhuoYun@PRAHS.com  
jzhuo@KitePharma.com