Paper SAS4530-2020

Troubleshooting Encoding Issues When Integrating Data from Multiple Systems

Lydia Sbityakov, Breanna Barton, SAS Institute Inc.

ABSTRACT

Symptoms, diagnoses, and cure for encoding issues: When integrating multiple data sources in a real-world environment, especially with legacy databases, it is not uncommon to encounter data sets with different encodings. What types of errors can this cause? How can you recognize that what you are seeing is a result of mixing various encodings? What are some of the methods that can be used to determine what the native encoding of your data sources and SAS® session are? This session walks through troubleshooting a real-life example and its resolution.

INTRODUCTION

The material in this paper is adapted from work on a data integration project that used SAS programs to combine data from multiple data sources. Early in the project we realized that not all the data was appearing as expected in our SAS datasets. We had trouble pinpointing exactly what the root cause of the problem was and how to correct it. This was because there were multiple parts of our architecture that required changes so that our data could be imported correctly into its target database.

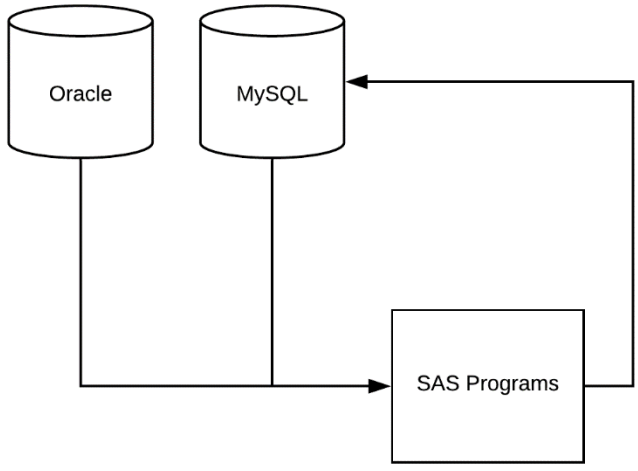

Figure 1 Data Integration Project Overview

BACKGROUND INFORMATION ON ENCODINGS

UTF-8 has become the standard encoding for most current files and systems because it can handle all international character sets and even emojis (😊). This is great for communicating with your clients and colleagues around the world but can be tricky when trying to combine UTF-8 data with data from older systems using encoding such as ASCII

(American Standard Code for Information Interchange), Windows-1252, or Latin-1 to name only a few common examples.

Older encodings can generally be converted to UTF-8 but it is not always possible to convert from UTF-8 to another encoding. Let's consider ASCII, which is a seven-bit encoding developed in the early 1960s. Descended from telegraph codes, it is the smallest commonly used character set with only 128 possible combinations of '0's and '1's. . This works fine for English since only 52 letters (including upper and lower case), numerals and a few characters for punctuation are needed.

UTF-8 characters take between one and four bytes so that multibyte characters will simply not fit in the limited space allocated for ASCII characters. Other encoding systems may run into the same problem.

Here is a table showing how UTF-8 characters are constructed:

Number of Bytes Character Sets	Byte 1 ASCII, Western European Characters	Byte 2 Russian, Greek, Arabic, Hebrew, African	Byte 3 Chinese, Japanese, Korean, Indic	Byte 4 emojis
1	0xxx xxxx			
2	110x xxxx	10xx xxxx		
3	1110 xxxx	10xx xxxx	10xx xxxx	
4	1111 0xxx	10xx xxxx	10xx xxxx	10xx xxxx

Figure 2 UTF-8 Structure

When working with data, it is essential to understand the different encoding types and how they are affected within a SAS session. The encoding of a SAS session or SAS data set, along with the encoding of systems and machines all have effects on data.

How can one recognize that errors are a result of invalid or incompatible encodings? Here we will highlight the common indicators of these issues along with some possible solutions.

EXAMPLES OF ENCODING ERRORS

UNREADABLE CHARACTERS

Have you ever opened a file, only to find it is a jumbled, unreadable mess of characters? This text file, which is not displaying correctly, contains a Russian poem by A.S. Pushkin. While unreadable in ANSI format, when the encoding is changed to UTF-8 the appearance changes drastically.

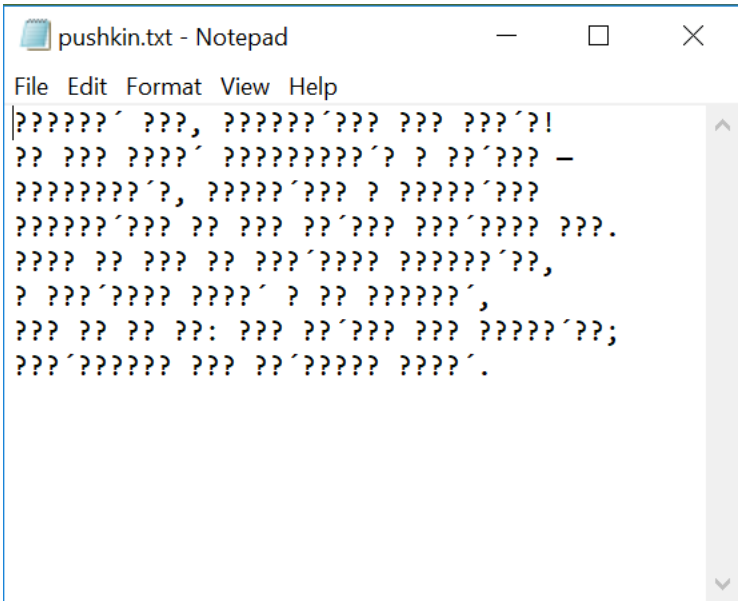

Figure 3 October 19th by A. C. Pushkin (ANSI encoding)

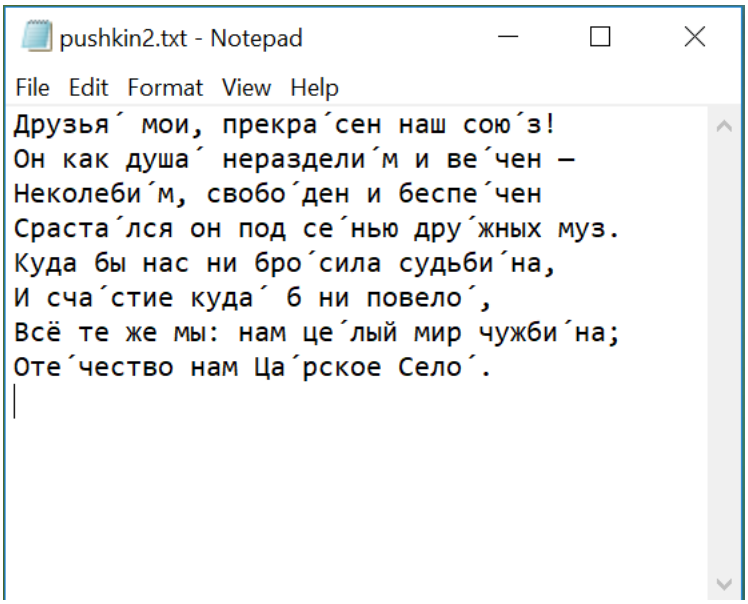

Figure 4 October 19th by A. C. Pushkin (UTF-8 encoding)

It is interesting to note that some of the characters are displayed correctly in the ANSI file. This includes punctuation marks like commas, periods, exclamation points, semi-colons, dashes, and spaces. This is because the encoding of these characters overlaps between ANSI and UTF-8.

In this example, it is easy to notice that something is amiss, but oftentimes the error will not be as extreme and only accent marks or certain letters will be incorrect.

WARNING MESSAGES IN YOUR SAS LOG FILE

When working with data from different environments, it is likely that the data will have different encodings and will need to be transcoded. Transcoding is the process of converting data from one encoding to another encoding.

Another potential encoding challenge that reflects this error is a warning message in the SAS log file. For example:

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

Output 1. SAS Warning Message

This message points to two possible problems. First, the characters do not exist in the target encoding and as a result cannot be displayed correctly. Second, the number of bytes per character is not the same between encodings. This runs the risk of truncating the data if the columns are not wide enough for the additional length.

UNEXPECTED LENGTH OF COLUMNS

When working with different systems, it is possible for data attributes to change as a result of different encodings. For example, the specified width on a table may change to try to help accommodate for changes in length during transcoding.

TABLES NOT APPEARING IN ENTERPRISE GUIDE

Other times, it isn't as easy to recognize that errors or warnings are a result of encoding discrepancies. For example, when connecting to a library in SAS, it is possible that although the connection is made successfully, the tables might not appear, or they may appear with different table attributes than expected. These can both be signs of mis-matched encodings.

STEPS TO RESOLVING ENCODING ISSUES IN SAS

ATTEMPT TO UNDERSTAND YOUR ENVIRONMENT

Encodings and correlated language settings are set all over the place. The SAS session has an encoding, and each data set will have an encoding. In addition, other environments or servers that are connected will have their own encodings (i.e. Oracle, MySQL connections) that may not align with the others.

Before attempting to change encodings, it is critical to find the current encoding for all environments, including the SAS session and data sets.

Here is a diagram that includes the default encodings for SAS sessions:

Default SAS Session Encoding Values

Operating Environment	Default ENCODING= Value	Description
z/OS	OPEN_ED-1047	OpenEdition EBCDIC cp1047-Latin1
UNIX	Latin1	Western (ISO)
Windows	WLatin1	Western (Windows)

Figure 5 Default SAS Session Encoding Values

To find the encoding for your SAS session:

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

```
 ENCODING=LATIN1   Specifies the default character-set encoding for the SAS session.
NOTE: PROCEDURE OPTIONS used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
```

Output 2 Session Encoding Output

To find the encoding for your SAS data set:

```
proc contents data=libname.dataset; run;
```

The CONTENTS Procedure

Data Set Name	WORK.DATA1	**Observations**	3
Member Type	DATA	**Variables**	19
Engine	V9	**Indexes**	0
Created	02/13/2020 09:31:55	**Observation Length**	441
Last Modified	02/13/2020 09:31:55	**Deleted Observations**	0
Protection		**Compressed**	CHAR
Data Set Type		**Reuse Space**	NO
Label		**Point to Observations**	YES
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64	**Sorted**	NO
Encoding	utf-8 Unicode (UTF-8)		

Figure 6 SAS Dataset Encoding Output

If working in a Linux environment, it may be easier to check file encodings via command line.

To find the encoding for your SAS data set:

```
>file -bi <filename>
```

This will return something similar to the following, which informs us that it is UTF-8, for example:

```
>text/plain; charset=UTF-8
```

After running these tests, we can see that the SAS session has latin1 encoding while our data set is using UTF-8 encoding. This could be a potential cause for other problems.

After checking SAS encodings, it is essential to check encodings for the other systems you are integrating with. For example, if working with an Oracle database, we should check the encoding for the tables that are being used.

To check encoding in an Oracle database:

```
select * from nls_database_parameters where parameter =
'NLS_CHARACTERSET';
```

If it is necessary to change the SAS session encoding, it can be changed using the ENCODING= option at the beginning of the session. It is important to note that the session encoding must be set at the beginning and cannot be changed throughout.

USE THE APPROPRIATE VERSION OF SAS EXECUTABLE

Whether from the command line or Enterprise Guide, when you run a SAS program either a single byte (ASCII) or UTF-8 version of SAS is executing the code. This is something that you can control easily from the command line. If you use Enterprise Guide, you can set your workspace server to use the UTF-8 version instead of the older ASCII version.

If you work in a complex, shared environment with many developers another option is to create separate workspace servers that run with different encodings.

If using Enterprise Guide, the workspace servers can be set up to toggle between servers manually. You may be on a system that is set up for older ASCII characters and will need to be re-programmed for UTF-8. We had to set up a separate workspace server that was UTF-8.

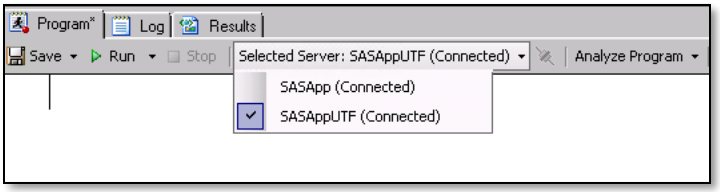

Figure 7 Screenshot of Enterprise Guide Workspace Server Options

Another option is running in batch; in batch, you can specify the encoding and execute the code on a SAS session with the stated encoding.

You may have two versions of the SAS executable.

One for ASCII:

```
/sso/sfw/sas/940/SASFoundation/9.4/sas
```

And another for UTF-8:

```
/sso/sfw/sas/940/SASFoundation/9.4/bin/sas_u8
```

When using the UTF-8 capable version, use the command's encoding option:

```
sas_u8  -encoding utf-8 program.sas
```

USE APPROPRIATE LIBRARY OPTIONS

There are LIBNAME statement options that we found essential for remedying the encoding problems during the process of integrating data: INENCODING/OUTENCODING and DBCONINIT.

INENCODING and OUTENCODING are library options that are specified in the LIBNAME statement. These options are applied to data sets that are coming into or out of a library.

The INENCODING= and OUTENCODING= LIBNAME options are best used when working in a SAS library with multiple encodings or for transcribing data that has a different encoding than the SAS session.

For example, one option could be used to read a wlatin1 data set in a UTF-8 session:

```
libname prct ‘c/td/data’ inencoding= ='wlatin1';
```

DBCONINIT addresses a separate issue. When connecting to MySQL tables in SAS, it is possible that although the libname is correctly assigned the tables are not visible in the SAS library. This can be remedied using the DBCONINIT libname statement option:

```
libname mydb mysql SERVER='td-cz' PORT=2940 DATABASE=testdata USER=admn
PASSWORD=xxxxxxxxxxxxx DBCONINIT="set names ‘utf8’";
```

The DBCONINIT command can be used to execute a user-defined initialization command immediately after every connection to the database. Any valid database command can be used, given that it does not result in a return parameter. In this case we can use "set names 'UTF8'" to ensure that the tables are encoded correctly and will be visible in our SAS library. This option is especially useful when working with other environments that are encoded differently.

Another option, which is not a library option, is the ENCODING= option that can be used in PROC or DATA steps. This is yet another option to help remedy a situation where data is garbled because it has a different encoding from what was used to create it. It is important to note that this option should not be confused with the ENCODING= option that is used to set the SAS session encoding.

RESOLVING LENGTH QUESTIONS MAY REQUIRE THE CVP ENGINE

The CVP (Character Variable Padding) engine is used to pad the length of variables. This is especially useful during transcoding as it helps prevent data truncation. The CVP engine will multiply the column length by 1.5 by default. It is possible to change the multiplier using the CVPMULTIPLIER= option.

When transcoding data from LATIN1 to UTF-8 it is likely that some characters that were previously 1 byte will need 1,2 or 3 bytes to hold the information. It is essential to change the width of the columns to ensure that the column is wide enough.

CONCLUSION

By the end of our project we realized that the system we started out with was really like the updated diagram below. The encoding of the MySQL database changed during the project from a 3-byte version of UTF-8 to a 4-byte version.

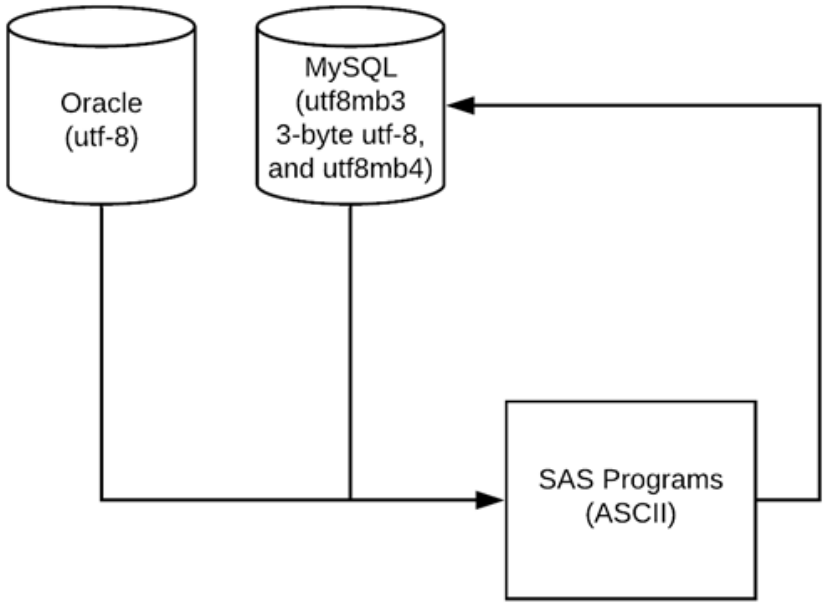

Figure 8 Updated Data Integration Project Overview

By adding a UTF-8 workspace server to our Enterprise Guide application, calling the UTF-8 version of the SAS executable in our batch scripts, and setting the appropriate library options on our connection strings to the external databases, we were able to successfully prepare our data extracts to upload into the target database.

However, not all encoding questions have an easy answer. In the case of the Russian poem, although the characters appeared much better with the encoding set to UTF-8, they are still not entirely correct.

This version of the poem was meant to include accents to help foreign readers and young students pronounce the words correctly.

Cyrillic characters with accents are not included even in UTF-8 and do not display correctly in our text file. They appear as apostrophes next to the letters they modify instead of above them.

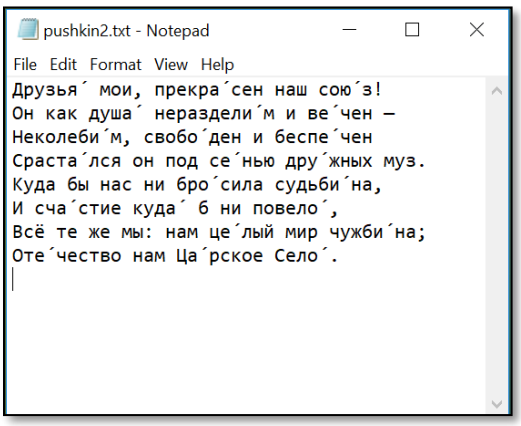

Figure 9 October 19th by A. C. Pushkin (UTF-8 encoding)

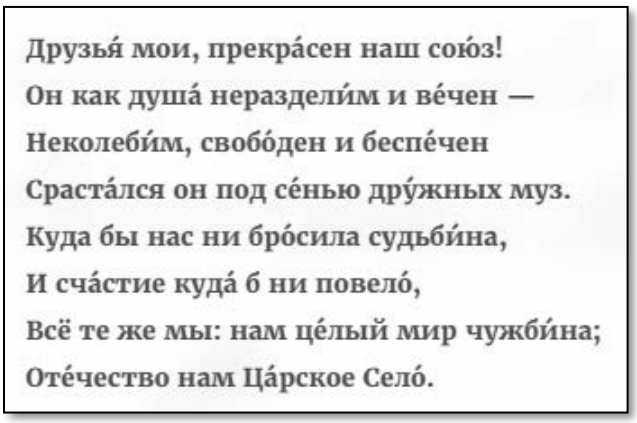

Figure 10 October 19th by A. C. Pushkin (html)

Special HTML programming was necessary to allow the correct display of accented characters:

```
<p>п<span class='letter letter-u'>у<span class='accent'>&acute;
```

UTF-8 handles most cases but there are still examples like this that can cause issues in your work.

REFERENCES

Bales, E and Zheng W. 2017. "SAS and UTF-8: Ultimately the Finest. Your Data and Applications Will Thank You!" Available at https://support.sas.com/resources/papers/proceedings17/SAS0296-2017.pdf

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

Edney, Shawn (2019, Sep 6). personal communication.

Pushkin, Alexander. "October 19". http://russianpoetry.yale.edu/poet/pushkin.html.

SAS 9.4 National Language Support (NLS): Reference Guide, Fifth Edition

Some NLS characters with egonek not correctly translated in a UTF8 session using Access to MySQL. Retrieved from SAS CUSTOMER SUPPORT.

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

CONTACT INFORMATION

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

Lydia Sbityakov
SAS Institute Inc.
Lydia.Sbityakov@sas.com

Breanna Barton
SAS Institute Inc.
Breanna.Barton@sas.com