
SAS®
GLOBAL
FORUM
2020
MARCH 29 - APRIL 1
WASHINGTON, DC
USERS PROGRAM

Your Data Will Go On: Practice for Character Data Migration

Edwin (You) Xie
SAS Institute Inc.

Your Data Will Go On: Practice for Character Data Migration

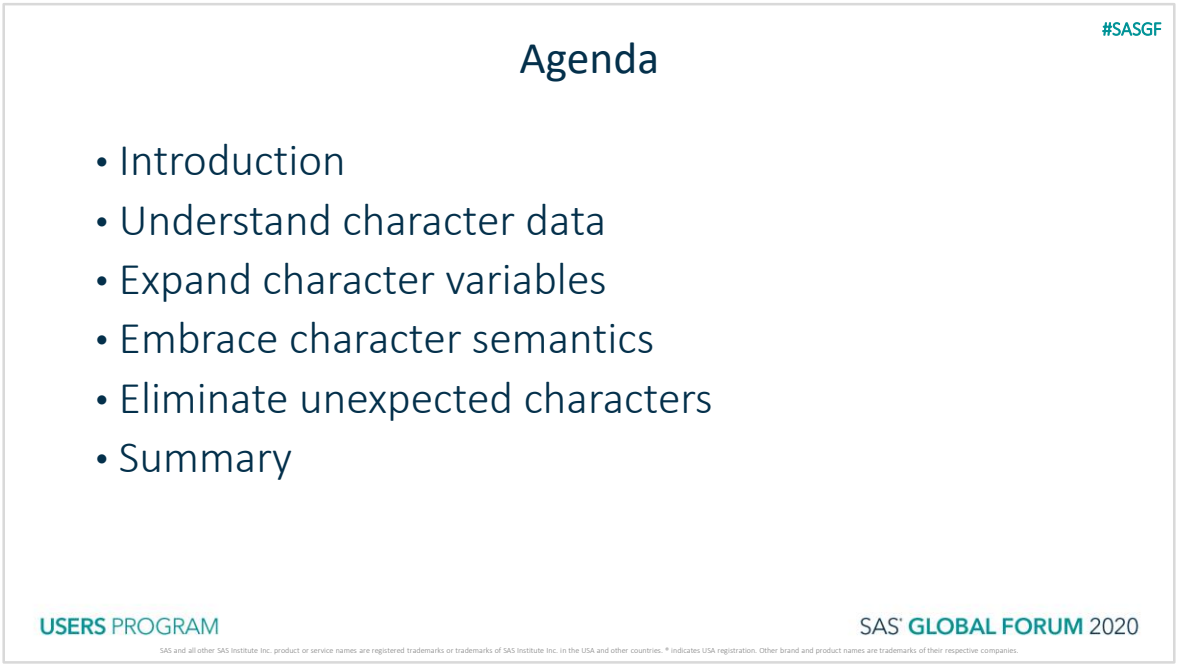
Edwin (You) Xie

Edwin Xie is a senior software developer in SAS Beijing R&D. He focuses on the globalization of SAS products. He is also supporting SAS Character Variable Padding engine and SAS Time Zone.

USERS PROGRAM

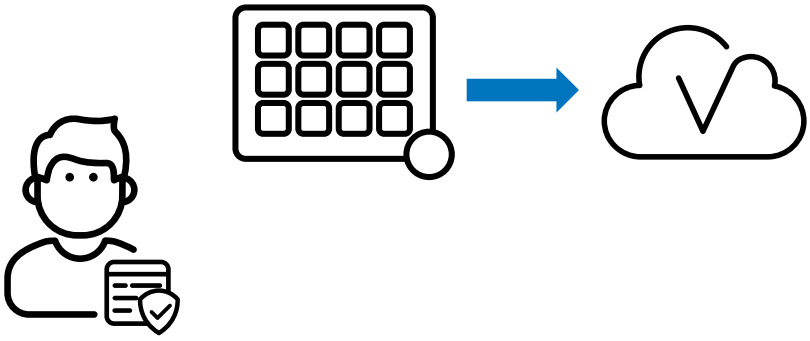
SAS® GLOBAL FORUM 2020

- Hello, every! I am Edwin from SAS Beijing R&D, I am focusing on globalization development for more than 8 years. In this session, I will talk about the data migration for SAS character data.



- This is my agenda
- First, I will use an example to demonstrate the common issues in character data migration
- Then, I will introduce some essential knowledge of character data to help you understand them.
- By understanding character data, I will use some examples to show you how to resolve these issues from 3 aspects
- Last, I will give you a summary

Introduction

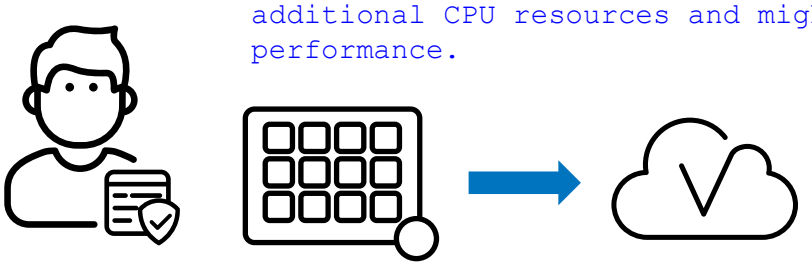


- First, let's see an example.
- Smith is working for a global company. He wants to migrate his existing data sets to SAS Viya.

#SASGF

Introduction

```
NOTE: Data file BANKLIB.CLASS.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.
```

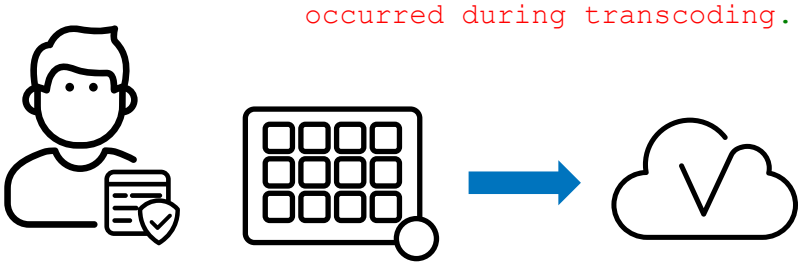


USERS PROGRAM SAS® GLOBAL FORUM 2020

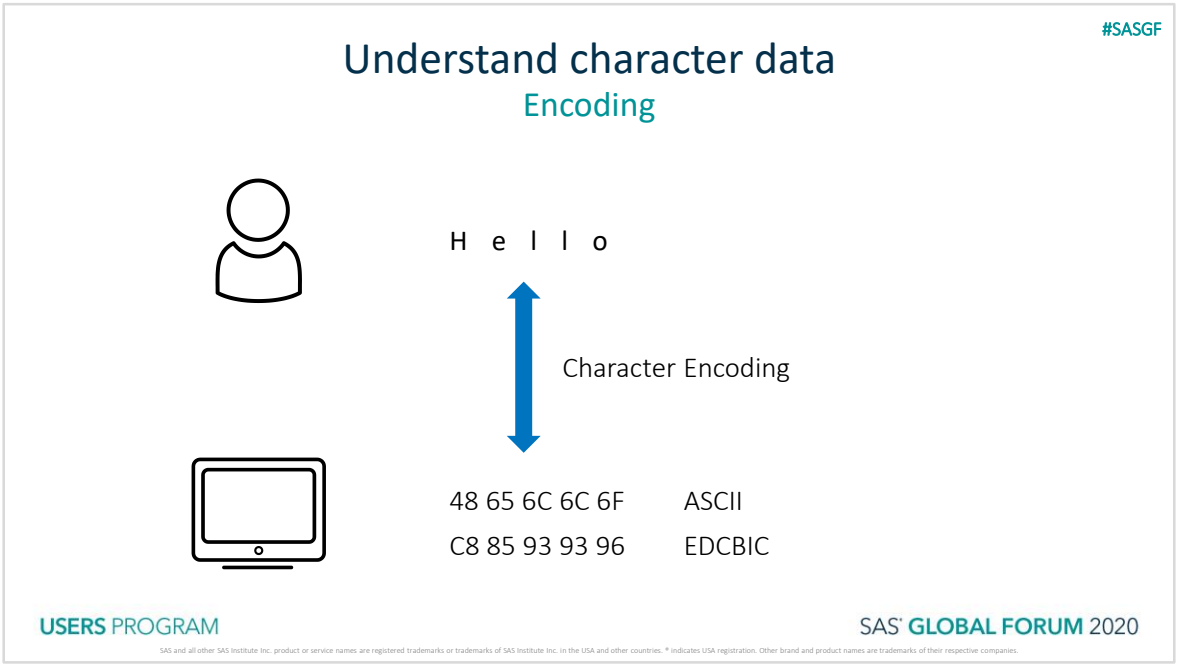
- During the data migration, he sees a note message like this. He has little knowledge about how the Cross Environment Data Access works. But a little performance decrease is acceptable for him, so he does not think the message is an issue for his migration.

Introduction

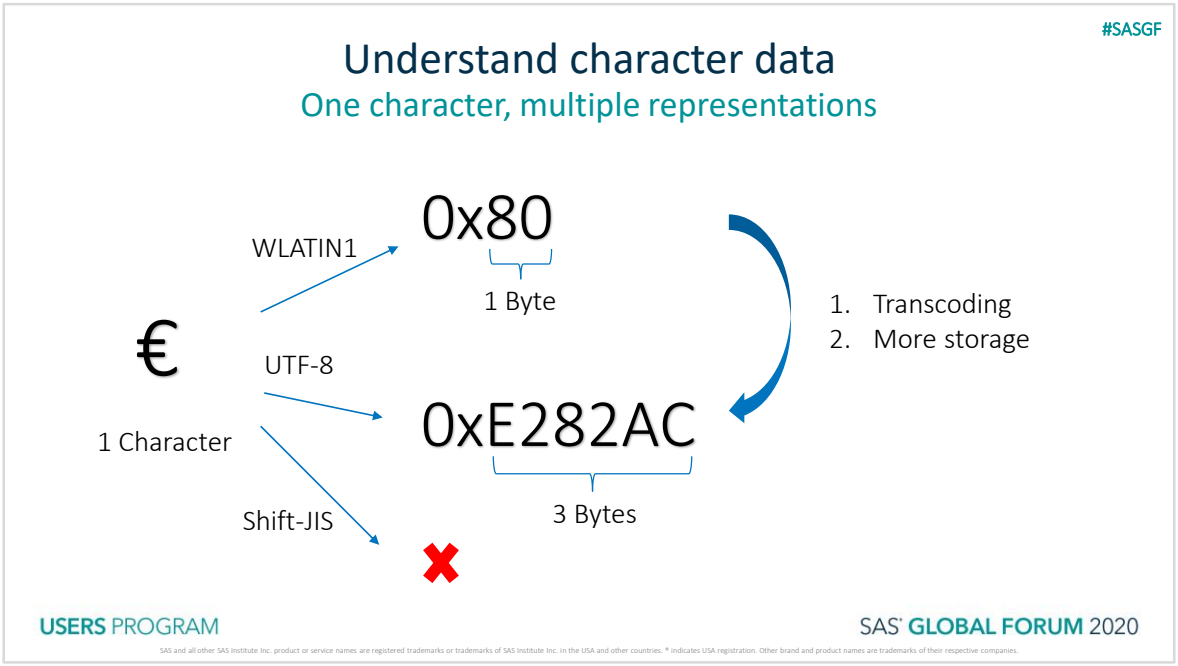
```
WARNING: Some character data was lost during
         transcoding in the dataset
         BANKLIB.CLASS. Either the data contains
         characters that are not representable
         in the new encoding or truncation
         occurred during transcoding.
```



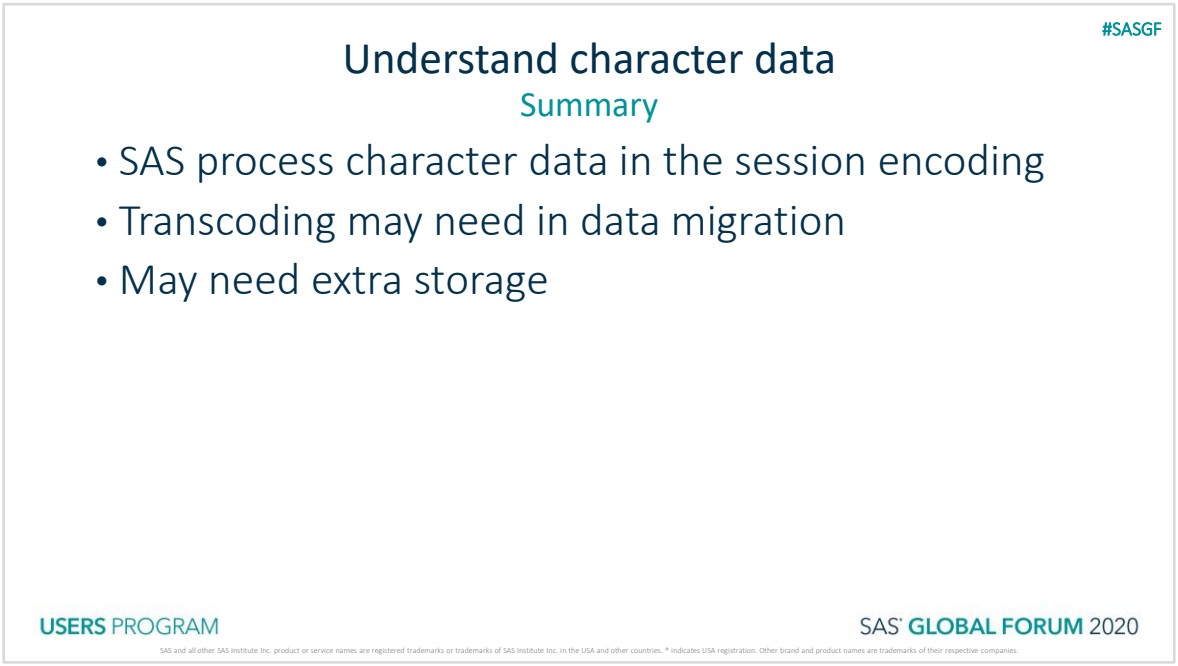
- However, he gets into trouble when he sees the warning message. Then he checks the migrated data set. Unfortunately, some strings are truncated, and he has no ideas about how to solve the truncation issue.



- To know why Smith's data is truncated, we need to understand character data first.
- As we know, computers use binary to store data. But how to map characters to binaries? That needs encodings. A character encoding defines a set of characters and their binary code.
- Only by knowing the encoding, we can identify the characters stored in a computer.
- Accordingly, If the encoding changed, the binary data to represent the same text may also change.



- Let us see an example. Here is a Euro sign on the left side. The character has different representations in different encodings.
- If the Euro sign is stored in a wlatin1 data set, it's code is eighty in hexadecimal.
- And if the encoding is Utf-8, it needs 3 bytes to store it.
- The character can not exist in a Shift-JIS data set, because the Shift-JIS character set does not contain the character.
- So if we open a data set whose encoding is different from the session encoding, transcoding may need.
- For example, If we open a wlatin1 data set in a UTF-8 environment, such as in a Unicode server, the data set need to be transcoded to UTF-8. Usually, the cross-environment data access, or CEDA for short, will be automatically used during the data access. That is why Smith sees the note message in our first example.
- Additionally, the same character data may need more storage in UTF-8. If the variable is not large enough to hold the transcoded data, the data will be truncated. That is why Smith sees the warning message in the example.



- That is the characteristic of SAS character data. Let us have a summary
- SAS usually processes character data in the session encoding.
- If the data set encoding differs from the session encoding, transcoding may be needed.
- The same character data may need more storage in the target encoding. So the original variables may not large enough to hold all the transcoded characters, and truncation may happen.

#SASGF

Expand character variables

The Data Truncation Issue

Variables in Creation Order				
#	Variable	Type	Len	Format
1	Name	Char	8	$8.
2	Sex	Char	1	
3	Age	Num	8	BEST2.

Obs	Name	Sex	Age
1	André	M	14
2	Valérie	F	13
3	Béatrice	F	13

Latin1

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

Obs	Name	Sex	Age
1	André	M	14
2	Valérie	F	13
3	Béatric	F	13

UTF-8

- If you encounter the truncation issue like Smith does, how to resolve it? Let see another example.
- Here is a data set, it records students' information for an international class. The data set contains 2 character variables Name and Sex and a numeric variable: Age.
- The data set is in Latin1 encoding. Now suppose you want to migrate the data set to another SAS environment which encoding is UTF-8
- When you open the data set directly in the new environment, you will see a warning message tell you truncation occurred.
- In the transcoded data set, as you can see, the last character 'e' is truncated. That's because the length of the Name variable is 8, which means it can store 8 bytes at most.
- In the latin1 data set, each character uses only one byte, so the variable can contain all the 8 characters. But in UTF-8, the e-acute character uses 2 bytes, so the whole word needs 9 bytes. But the variable length is still 8, so the last character is truncated.

#SASGF

Expand character variables

CVP

```
LIBNAME mylib CVP 'path to library';
```

Variables in Creation Order

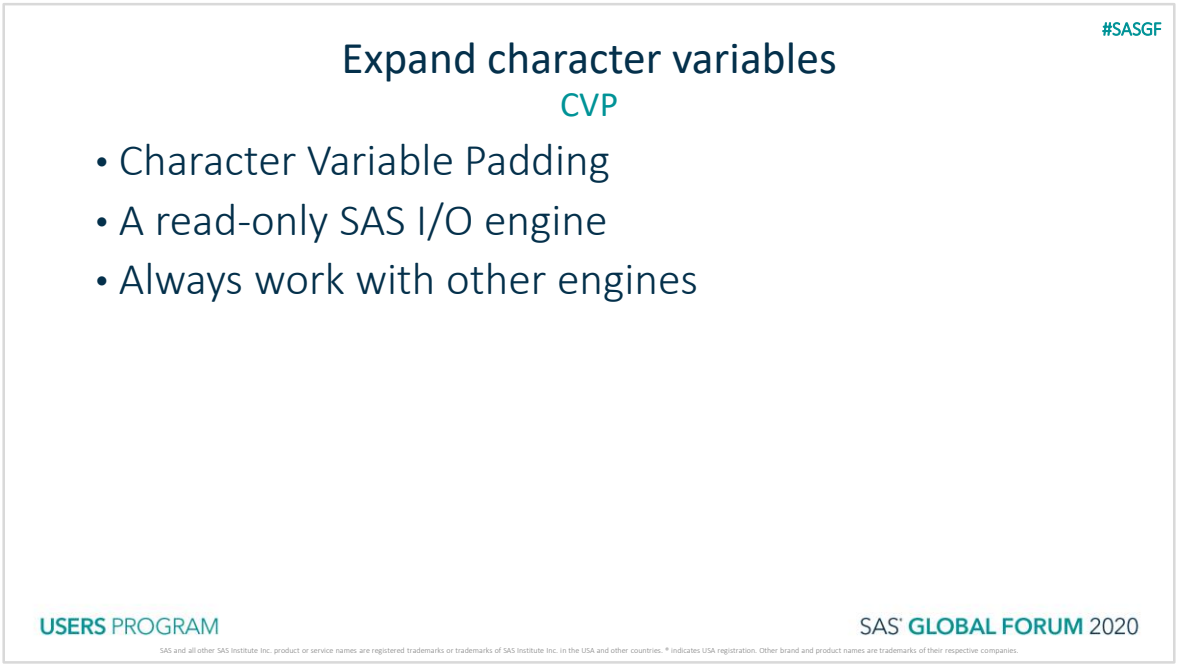
#	Variable	Type	Len	Format
1	Name	Char	16	$16.
2	Sex	Char	2	
3	Age	Num	8	BEST2.

Obs	Name	Sex	Age
1	André	M	14
2	Valérie	F	13
3	Béatrice	F	13

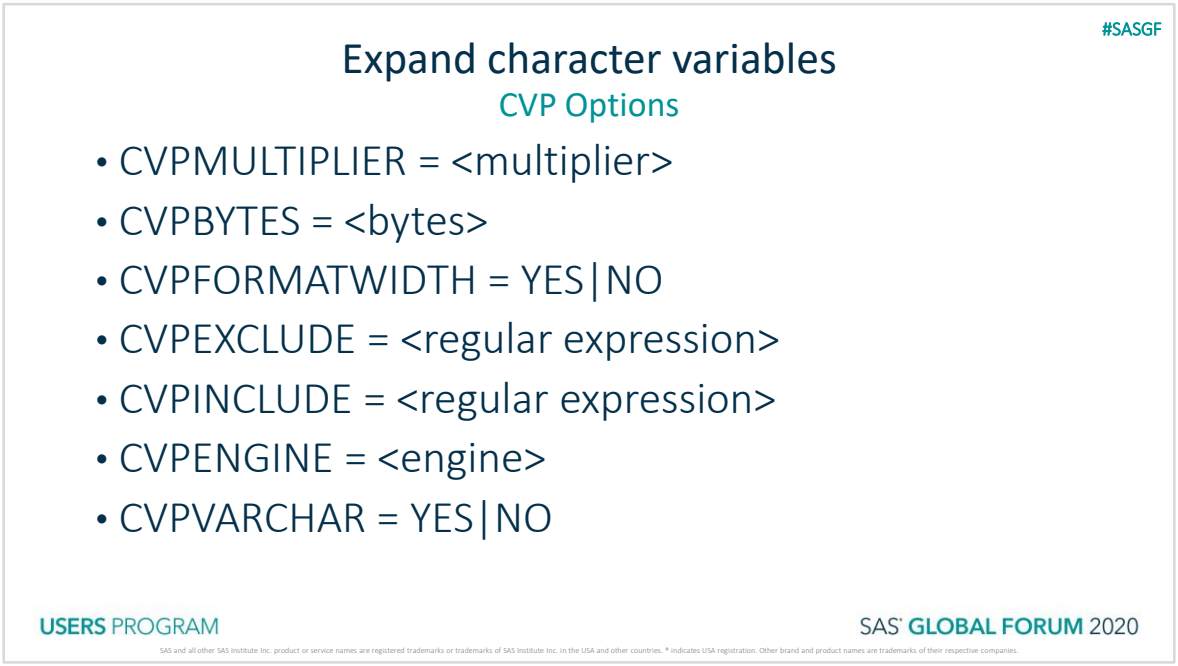
USERS PROGRAM

SAS GLOBAL FORUM 2020

- A candidate solution is to expand the character variables before the transcoding. SAS provides several tools to help you do that.
- CVP engine is the first tool I want to introduce to you. To use the CVP engine, just add the keyword CVP in the libname statement, then open the data set again.
- You can see, the length of the variables are expanded twice, as well as the character formats. Meanwhile, the numeric variable and format keep unchanged.
- To print the data set again with CVP engine, the string is not truncated anymore.



- Let's have a look at the CVP engine. CVP standards for Character Variable Padding, the engine is initially designed to expand SAS character variables
- It is a read-only engine. That means it does not save modifications back to the data set. If you want to keep the changes, you need to create a permanent copy of the modified data set.
- The engine relies on other engines to access the data file. By default, the BASE engine is used. It can also work with other engines by using options.



- Here are the available options for the CVP engine.
- The CVPMULTIPLIER and CVPBYTES specify how much to expand the variables. For CVPMULTIPLIER, you can also use the value 0 to let the CVP engine choose the multiplier automatically.
- The CVPFORMATWIDTH is used to expand character formats
- The CVPEXCLUDE and CVPINCLUDE are used to select variables. Both of them support regular expressions
- The CVPENGINE is used to specify the underlayer engine used to access the data files. The default engine is the BASE engine as I just mentioned.
- The CVPVARCHAR is used to convert CHAR data type to VARCHAR. I will introduce it later.
- You can also refer to my paper for the details of these options.
- Here is all about using the CVP engine to expand character variables.

#SASGF

Expand character variables

CAS Options

- System options CASNCHARMULTIPLIER=
- LIBNAME option NCHARMULTIPLIER=
- Data Set option NCHARMULTIPLIER=

- Take effects when the client encoding is not UTF-8
- The latter ones overwrite the previous ones.
- Default value
  - 1 for SBCS: latin, wlatin, …
  - 1.5 for DBCS: euc-cn, shift-jis, …

USERS PROGRAM SAS® GLOBAL FORUM 2020

- Besides the CVP engine, if you are working in a SAS Client, and want to load your data to CAS. You can also use these 3 options to expand the character variables.
- All the 3 options provide similar functionality with different scope. The first one takes effect for all libname engines, the second one works for one libname assignment, and the last one is for one data set. If multiple options are specified at the same time, the below ones override the above ones.
- The default values are one or one point five based on the client's encoding.
- There is one thing to note here, these options take effect during transcoding. That means they only take effect when the client encoding is different from the CAS session encoding which is UTF-8.

#SASGF

Expand character variables

CAS Options

```
libname sys cas;
libname lib cas NCHARMULTIPLIER=1.5;

data sys.class_sys;
   set class;
run;

data lib.class_lib;
   set class;
run;

data lib.class_ds(NCHARMULTIPLIER=2);
   set class;
run;
```

Override the system option

Error: data truncation

character variables * 1.5

Override the libname option
character variables * 2

Obs	Name	Sex	Age
1	André	M	14
2	Valérie	F	13
3	Béatrice	F	13

USERS PROGRAM

SAS GLOBAL FORUM 2020

- Let us see an example. Suppose the client encoding is latin1. And the input data set is the class data set we used in the previous example.
- If no options are specified, the system option will be used. Since latin1 is a single-byte character set, the default multiplier is one. Which means no variables will be expanded. So a truncation will happen.
- The second data set uses the libname option, which value is one point five. It overrides the system option so all character variables will be expanded 1.5 times.
- The last data set uses the data set option, it overrides both the libname option and the system option, so the variables will be expanded twice.

#SASGF

Expand character variables

Transport file

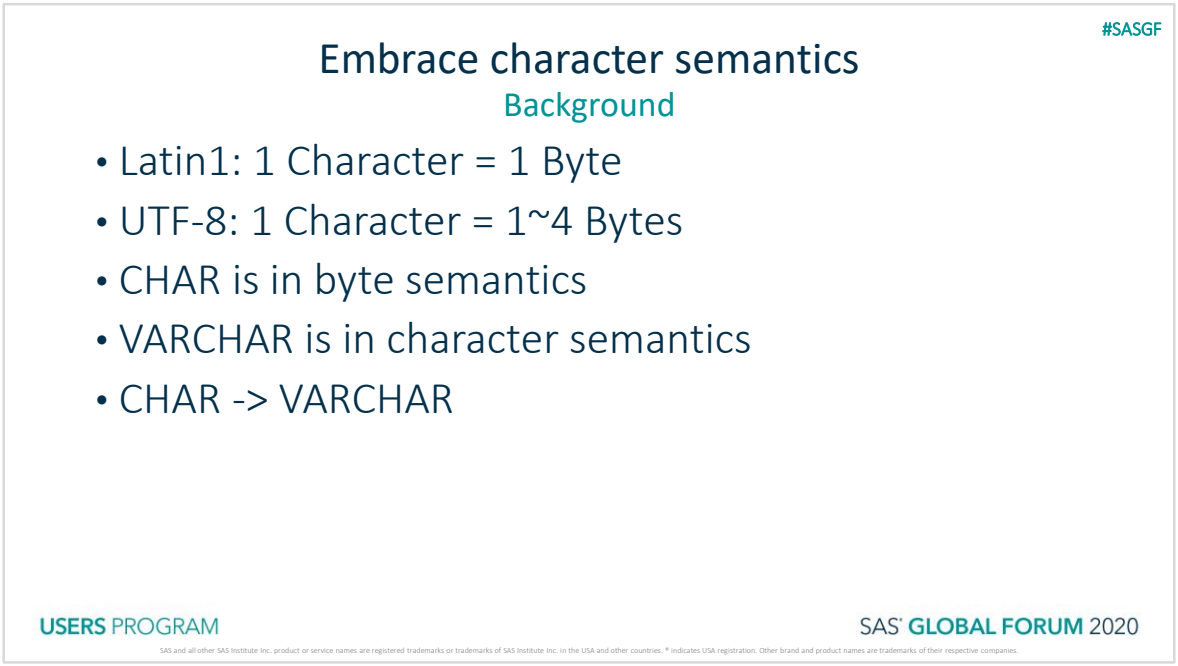
- PROC CIMPORT options
  - EXTENDVAR=multiplier | AUTO
  - EXTENDFORMAT=YES | NO

```
filename infile 'transport-file';
libname target 'SAS-data-library';
proc cimport infile=infile library=target extendvar=1.5;
run;
```

USERS PROGRAM

SAS GLOBAL FORUM 2020

- That is how to expand variables in SAS data sets.
- If your data is in SAS transport files, you can also do the similar expansion during importing them.
- Beginning from Viya 3.5, the cimport procedure provides 2 options to expand the character variables and formats in transport files.
- Here is an example of using the new option extendvar to expand all character variables and formats 1.5 times. Like the CVP engine, you can also use AUTO to let the procedure to choose the multiplier automatically.



- Above is about expand variables length, let us have a brief summary. During data migration, truncation happens because the same data requires more bytes in the new environment. So we use tools include CVP engine, CAS options and CIMPORT procedure options to expand character variables. These solutions focus on the potential bytes changing during data migration.
- From another point of view, the number of characters, not bytes, do not change during data migration. So if we use character semantics, which means the variables length are character length, truncation will not happen
- As we know, if a variable is defined as VARCHAR, it is in character semantics. So we can convert our existing character variables to VARCHAR type, and the CVP engine is the tool to help us do the conversion.

#SASGF

Embrace character semantics

CVP

```
libname mylib 'path' cvpvarchar=yes;
proc contents data=mylib.class; run;
```

Variables in Creation Order					
#	Variable	Type	Bytes	Chars	Format
1	Name	Varchar	32	8	$32.
2	Sex	Varchar	4	1	
3	Age	Num	8		BEST2.

USERS PROGRAM

SAS GLOBAL FORUM 2020

- Here is an example of using the CVP engine to convert CHAR to VARCHAR. By using the option CVPVARCHAR equal to YES, All the CHAR variables are converted to VARCHAR.
- A varchar variable has 2 length properties: byte length and character length. In the converted data set, the character length is equal to the original byte length. And the byte length is set to the maximum possible value based on the session encoding.
- In this case, the session encoding is UTF-8, so the byte length is 4 times of the character length because a UTF-8 character can use up to 4 bytes in SAS.
- The format length is also changed according to the variable's bytes length.

#SASGF

Embrace character semantics

Data Connector

Variables in Creation Order			
#	Variable	Type	Len
1	Name	Char->VARCHAR	8
2	Sex	Char	1 -> 2
3	Age	Num	8

- Load the data to CAS.
- Convert “Name” to VARCHAR.
- Double the length of “Sex”

- Besides the CVP engine, if you are loading data to CAS, you can also use the CAS data connector to do the similar conversion. Let’s still use the CLASS data set as an example.
- Assume we want to do 3 tasks at the same time: 1. Load the data set to CAS. 2. Convert the name variable from CHAR type to VARCHAR. 3. Keep the Sex variable as CHAR type, but double its size.

#SASGF

Embrace character semantics

Data Connector

```
proc casutil;
  load casdata="data/class.sas7bdat" /* path of data file */
       incaslib="CASUSER(userid)"    /* input caslib      */
       casout="class_cas"            /* CAS table name    */
       importOptions={ filetype="basesas",
                       VarcharConversion=2,
                       CharMultiplier=2 };
quit;
```

USERS PROGRAM

SAS® GLOBAL FORUM 2020

- By using CAS Data Connector, we can use only one statement to do all the 3 things. In this code, we use the Data Connector to load the CLASS data file to CAS. And specify how to convert the data set in the import options.
- The VarcharConversion equal to 2 means convert the CHAR variables which length is larger than or equal to 2 to VARCHAR type. So the Name variable which has a length 8 will be converted to VARCHAR. Meanwhile, the Sex variable with a length one will not be converted.
- The CharMultiplier equal to 2 specifies a multiplier 2. Which means expanded the length of all the CHAR variables twice. In this case, the variable Sex keeps in CHAR type, so its length will be doubled.

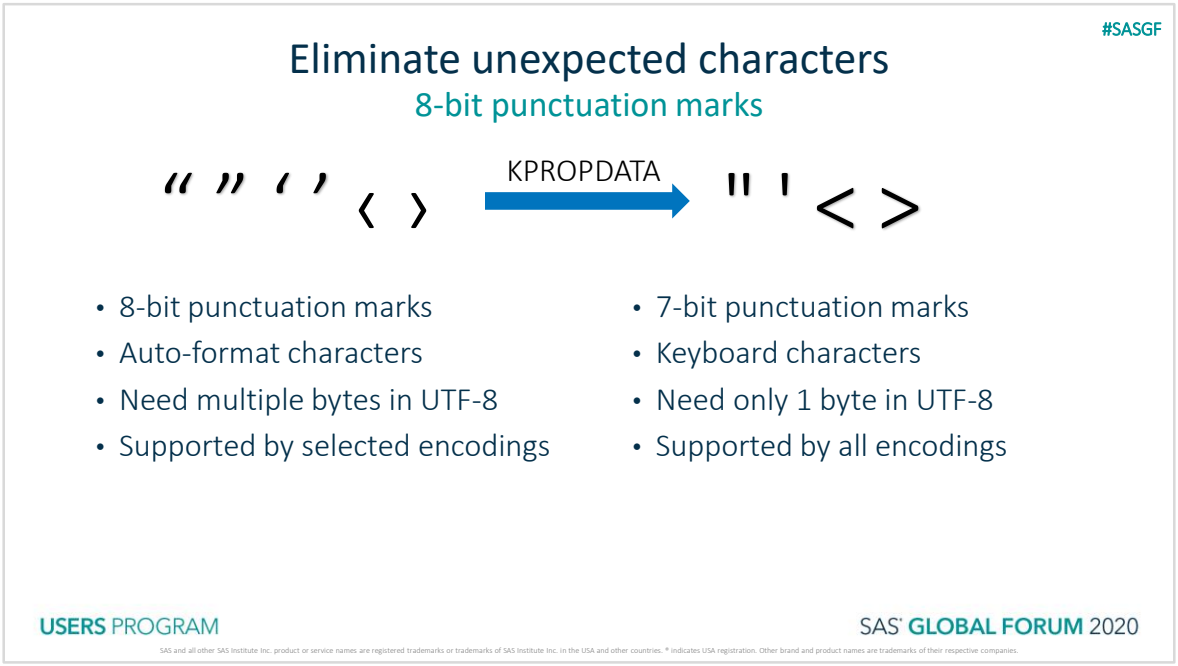
#SASGF

Embrace character semantics

Data Connector

Variables in Creation Order					
#	Variable	Type	Bytes	Chars	Max Bytes Used
1	Name	Varchar	32	8	8
2	Sex	Char	2		
3	Age	Num	8		

- Here is the result after the conversion. It is what we expected.



- During character data migration, we not only need to focus on the variables but also need to pay attention to the content. Because unexpected characters may exist in the data, and they may cause troubles.
- For example, if your character data is in English, you may feel your data will not be a problem during data migration. Because English characters and symbols are usually supported by commonly used encodings and share the same code points.
- However, in reality, you may still encounter troubles during data migration. Because the data may contain 8-bit punctuation marks, such as curly quotes. These marks are usually introduced by the applications with Auto format features such as Microsoft Word. These applications automatically replace 7-bit punctuation marks such as the straight quotes with 8-bit ones such as the curly quotes.
- These 8-bit characters complicate data migration. We can use KPROPDATA to convert them to 7-bit marks to smooth the data migration process.

#SASGF

Eliminate unexpected characters

8-bit punctuation marks

```
/* The session encoding is wlatin1 */
data mylib.notice;
   message="We will visit the “National Museum”
            next Friday";
   output;
   message=kpropdata(message, 'PUNC');
   output;
run;
```

USERS PROGRAM

SAS® GLOBAL FORUM 2020

- Here is an example. We have a data set that contains a notification message. The first observation is the original message which contains curly quotation marks. The second observation uses kpropdata with the ‘PUNC’ option to convert the curly quotes to straight quotes. These 2 observations have the same length in WLATIN1 encoding. However, if they are printed in UTF-8, you can see the difference.

#SASGF

Eliminate unexpected characters

8-bit punctuation marks

```
/* Open it in UTF-8 */
proc print data=mylib.notice; run;
```

Obs	message
1	We will visit the “National Museum” next Fr
2	We will visit the "National Museum" next Friday

USERS PROGRAM

SAS GLOBAL FORUM 2020

- In UTF-8, you can see the first observation is truncated. That because each curly quote needs 3 bytes in UTF-8, comparing to the straight quote which needs only one byte each.
- If we use KPROPDATA to convert all the punctuation marks to 7-bit ones, like the second observation, we can avoid the potential truncation during data migration.

#SASGF

Eliminate unexpected characters

Unprintable characters

- The binaries are not supported by the encoding
- Display as blank, question mark or garbage
- May cause unexpected result in string manipulation

USERS PROGRAM

SAS® GLOBAL FORUM 2020

- Until now, all the cases above are about how to prevent potential issues before or during data migration. But sometimes there is another situation that the data has already corrupted after data migration, and we do not want to port the data again.
- The corrupted characters, may unprintable or display as garbage, are hazards for your data, because they may cause unexpected results in string manipulation.
- Fortunately, we can also use KPROPDATA to remove or fix these characters.

#SASGF

Eliminate unexpected characters

Unprintable characters

```
data corrupted;
   length text $ 30;
   text = 'The SAS® System in UTF-8';
   output;
   text = 'The SAS' || 'ae'x || ' System in Latin1';
   output;
run;
```

A registered trademark in Latin1
But an unprintable character in UTF-8

USERS PROGRAM

SAS GLOBAL FORUM 2020

- Here is an example. The data set contains an unprintable character in its second observation. The hexadecimal ‘ae’ is invalid code in UTF-8.

#SASGF

Eliminate unexpected characters

Unprintable characters

```
/* The data is corrupted */
proc print data=corrupted; run;
```

Obs	text
1	The SAS® System in UTF-8
2	The SAS  System in Latin1

- If we print the data set, we will see an invalid character in the result.
- To avoid the invalid code cause unexpected issue and dirty our report, we can simply use KPROPDATA to remove it.

#SASGF

Eliminate unexpected characters

Unprintable characters

```
/* Remove the corrupted character */
data removed;
   set corrupted;
   text = kpropdata(text, 'TRIM');
run;
```

Obs	text
1	The SAS® System in UTF-8
2	The SAS System in Latin1

USERS PROGRAM

SAS GLOBAL FORUM 2020

- Here we use KPROPDATA with the 'TRIM' option and print the data again. You can see the invalid code in the second observation is removed.

#SASGF

Eliminate unexpected characters

Unprintable characters

```
/* Fix it */
data fixed;
   set corrupted;
   keep new;
   new = kpropdata(text, 'REMOVE');
   if new = ' ' then
      new = kpropdata(text, 'REMOVE', 'latin1');
run;
```

Remove the data string if any unprintable characters are found

Transcode it from latin1 to session

Obs	new
1	The SAS® System in UTF-8
2	The SAS® System in Latin1

- Furthermore, if we know the invalid code is a latin1 character, we can even fix it using KPROPDATA.
- In this example, we use the REMOVE option to indicate KPROPDATA to remove the whole text if it contains unprintable characters. So in the first KPROPDATA call, the function returns blank if the observation contains unprintable characters.
- Then, we can know which observation contains the invalid code by checking if the return value is blank.
- Since we already know the corrupted text is in latin1 encoding, we can further use KPROPDATA with the third input encoding option, to transcode the text from latin1 to the session encoding UTF-8. Now we successfully recovered the corrupted trademark character.

#SASGF

Summary

- Character data characteristics
  - Encodings & Transcoding
  - Byte & Character semantics
- Manage the character data
  - Expand character variables
  - Embrace character semantics
  - Eliminate unexpected characters

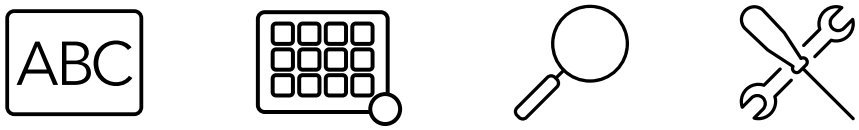
USERS PROGRAM

SAS® GLOBAL FORUM 2020

- Let's have a summary.
- In this session, first, we have a quick overview of the characteristics of character data. The same character may have different codes in different encodings. In some encodings, one character uses only one byte, while in some other encodings, one character may need multiple bytes. So if we transcode a character from one encoding to another, the byte length may change.
- Then, we learned the common issues during character data migration and how to handle them. We can expand variables length, convert CHAR variables to VARCHAR to use character semantics, convert 8-bit marks to 7-bit ones, and remove or fix unexpected characters in our data.

#SASGF

Summary



USERS PROGRAM

SAS® GLOBAL FORUM 2020

- To make sure the success of character data migration. First of all, you need to have knowledge of the characteristics of character data.
- Then you need to know your data and your target environment. Such as the encodings and the languages used.
- By knowing that, you can analyze the risks and potential issues during the data migration. At this stage, you can ask yourself some questions, such as “will the data be truncated? Will all the characters can be represented in the new environment?” And so on.
- Last to choose the best methods and tools to help you resolve these issues. Such as using the CVP engine to help your migration.
- Due to the time limitation, I can hardly cover every issue in data migration. However, by acquiring the knowledge I introduce in this session, I believe you can identify and solve the potential issues during character data migration yourselves. Once you are free to migrate your data to the latest platform, your data will be your most solid asset, along with the latest analytics technology, to ensure the success of your business. Thank you!

Thank you!

Contact Information
you.xie@sas.com

Reminder:

Complete your session survey in the conference mobile app.

USERS PROGRAM

SAS® GLOBAL FORUM 2020


SAS®
GLOBAL
FORUM
2020
USERS PROGRAM
MARCH 29 - APRIL 1 | WASHINGTON, DC | #SASGF