Paper 433-2017

Using International Character Sets with SAS® and Teradata

Greg Otto, Teradata Corporation

Austin Swift, Salman Maher, SAS Institute Inc.

ABSTRACT

If you run SAS® and Teradata software with default application and database client encodings, some operations with international character sets appear to work because you are actually treating character strings as streams of bytes instead of streams of characters. As long as no one in the "chain of custody" tries to interpret the data as anything other than a stream of bytes, then data can sometimes flow in and out of a database without being altered, giving the appearance of correct operation. But when you need to compare a particular character to an international character, or when your data approaches the maximum size of a character field, then you might run into trouble. To correctly handle international character sets, every layer of software that touches the data needs to agree on the encoding of the data. UTF-8 encoding is a flexible way to handle many single-byte and multi-byte international character sets. To use UTF-8 encoding with international character sets, we need to configure the SAS session encoding, Teradata client encoding, and Teradata server encoding to all agree, so that they are handling UTF-8 encoded data. This paper shows you how to configure SAS and Teradata so that your applications run successfully with international characters.

INTRODUCTION

Working with international character data can be a challenge because the data must be handled correctly at every step in its journey. Every layer of software that touches the data needs to agree on the encoding of the data as it moves between systems and applications.

The journey begins at the source. Is the data properly encoded when it is first encountered? Then the data must be correctly loaded into the database system. Finally, it must be handled correctly as it passes between Teradata and SAS for analysis. This paper follows the data along this journey, with guidance along the way, to help configure SAS and Teradata to work with international characters.

Most of the discussion in this paper focuses on UTF-8 encoding, which is a flexible variable-length encoding scheme. UTF-8 efficiently handles data based on the Latin alphabet, often requiring only one byte to represent each Latin character. It is also flexible enough to support international character sets and other new data including emoticons, by using multi-byte sequences of two to four bytes per character.

Three character set and encoding configuration settings must be considered as data moves between SAS and Teradata:

1. Teradata server character set - How the data is stored in the Teradata Database. If the data is not loaded correctly, then applications are never able to retrieve it in the form that is expected.

2. Client character set, or external encoding - The encoding used by the Teradata client software when data is transferred between Teradata and SAS.

3. SAS session encoding - How the data is encoded within the SAS system, including how it is stored in SAS data sets.

The following diagram illustrates how the components in an analytic environment must all agree on the character encoding to successfully load data to the database, store and use the data inside the database, and read data from the database. In this example, the data is in the Turkish language, encoded in Teradata LATIN (single-byte encoding) using the Turkish Cyrillic character set (latin5).

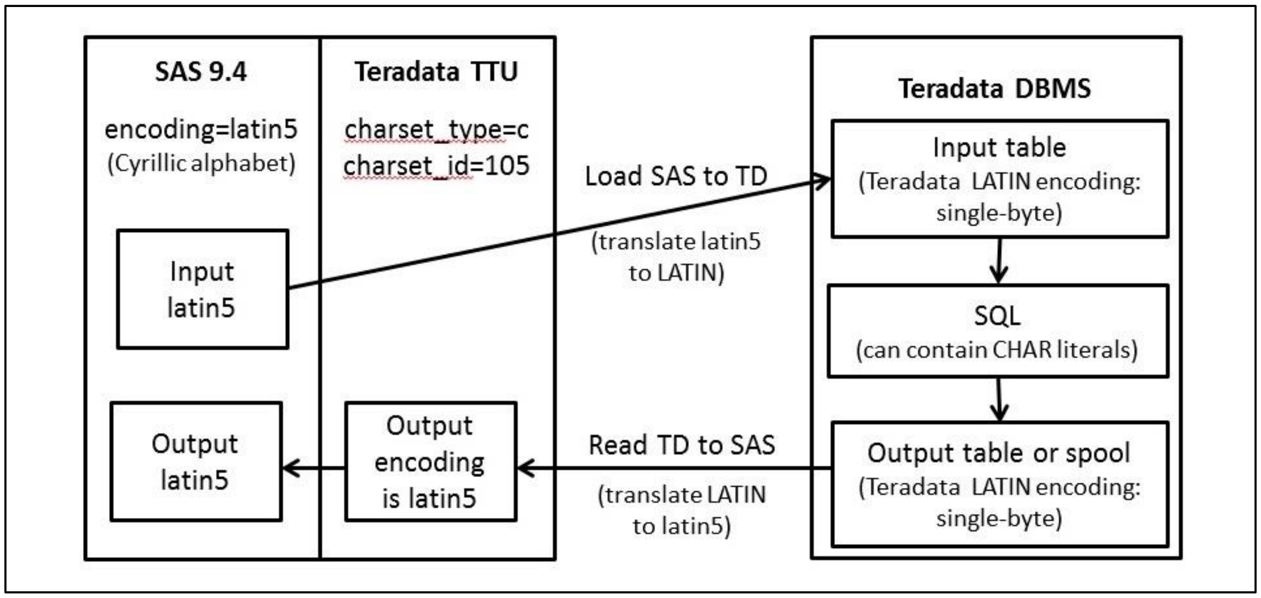

SAS 9.4
encoding=latin5
(Cyrillic alphabet)
Teradata TTU
charset_type=c
charset_id=105
Teradata DBMS
Input table
(Teradata LATIN encoding: single-byte)
Load SAS to TD
(translate latin5 to LATIN)
Input latin5
SQL
(can contain CHAR literals)
Output latin5
Output encoding is latin5
Read TD to SAS
(translate LATIN to latin5)
Output table or spool
(Teradata LATIN encoding: single-byte)

Figure 1. Using SAS/ACCESS to Teradata to Work with Teradata LATIN Data

TERADATA SERVER CHARACTER SETS

The Teradata database supports two primary server character sets: Teradata LATIN and Teradata UNICODE. (Other legacy server encodings are available in the Teradata database, but we do not address them in this paper.)

TERADATA LATIN

The Teradata LATIN character set is a single-byte encoding. It is an extension of the R2 international character set that is designed to support both latin1 and latin9 character sets.

Teradata LATIN supports languages that are based on the Latin alphabet, including English and many European languages.

Teradata LATIN encoding is designed to support more than one single-byte external encoding. This means it cannot exactly represent all client (external) single-byte encodings. Some characters are not supported, depending on which client encoding is used. See the Teradata internationalization documentation for specific limitations depending on your client encoding. The Teradata documentation includes translation tables for each single-byte client encoding.

Character set LATIN should only be used to store data that can be accurately translated to Teradata LATIN. It should not be used to store untranslated data.

TERADATA UNICODE

The Unicode character set is a standard two-byte (16-bit) encoding of virtually all characters in all current world languages, including single- and multi-byte client character sets.

Unicode is a canonical character set. Data stored in Unicode encoding can be shared among heterogeneous clients. The Teradata database supports characters from Unicode 6.0.

For a list of the supported characters, see the Unicode Server Character Set document. For additional characteristics of the Unicode Server Character Set, see the Teradata SQL: Data Types and Literals manual. (Teradata currently does not support surrogate code points (4-byte UTF-16) defined in Unicode 3.1 or above.)

Table 1 lists the various sections of the Unicode encoding scheme.

Section	**Languages Supported**
General scripts	Latin, Greek, Cyrillic, Hebrew, Arabic, Indic, and other characters
Symbols	Arrows, mathematical symbols, and punctuation
CJK Phonetics and Symbols	Hiragana, Katakana, and Bopomofo (CJK stands for Chinese, Japanese, and Korean)
CJK Ideographs	Chinese, Japanese, and Korean ideographs
Hangul Syllables	Complete set of modern Hangul.
Surrogates	Code points that extend the range of Unicode within ISO 10646
Private Use Area	Site-defined characters from KanjiEBCDIC, KanjiEUC, and KanjiShift-JIS
Compatibility Zone	The Compatibility Zone contains half-width and full-width variants of characters defined by Japanese standards
Emojis and other Ideographs	To work with data outside of the characters defined in the Unicode 6.0 specification, use the Unicode Pass Through (UPT) feature in TD 16.00

Table 1. Unicode Character Set Sections

SETTING THE TERADATA SERVER CHARACTER SET FOR DATA TABLES

The Teradata server character set is determined in the following order:

1. Column character set, as specified in the table DDL when the table was created.
2. User default, as defined in the user profile. Each Teradata user has default character set, LATIN or UNICODE, depending on how the user was created by the Teradata administrator.
3. System default, as defined by the system language mode when the database was initialized.

To create tables in Teradata using the UNICODE Server Character Set when the system or user does not default to UNICODE, add the server character set definition to the table DDL for each character column.

In this example, a simple table is copied from the SAS Work library to Teradata using the UNICODE Server Character Set. The SAS data set option DBTYPE modifies the CREATE TABLE statement that is passed to Teradata by adding the character set to the data type for column “uc”:

```
options set sastrace=',,,d' sastraceloc=saslog no$stsuffix;
data some_char_data;
length uc $10;
input uc;
cards;
Line1
Line2
Line3
; run;
libname td teradata server=tdbms user=grotto pwd=******;
data td.test_unicode_ddl (dbtype=(uc="CHARACTER SET UNICODE"));
  set some_char_data;
run;
```

```
TERADATA_1: Executed: on connection 2
CREATE MULTISET TABLE "test_unicode_ddl" ("uc" VARCHAR(10) CHARACTER SET
UNICODE);COMMIT WORK

NOTE: The data set TD.test_unicode_ddl has 3 observations and 1 variables.
```

Output 1. Teradata CREATE TABLE Statement with CHARACTER SET UNICODE

A complete Teradata table DDL can also be specified within a SAS program using the SQL procedure. See the SAS/ACCESS documentation for more details.

CHECKING THE DATA IN TERADATA

Some Teradata SQL statements and functions are very useful for checking the data in the database to make sure it is stored correctly.

SHOW TABLE

Use SHOW TABLE to examine the Teradata table definition, for the correct CHARACTER SET definition for each character column in the table:

```
SHOW TABLE grotto.test_unicode_ddl;
```

```
CREATE MULTISET TABLE grotto.test_unicode_ddl
(
      "uc" VARCHAR(10) CHARACTER SET UNICODE CASESPECIFIC
) PRIMARY INDEX ( "uc" );
```

Output 2. SHOW TABLE with UNICODE Column

CHAR2HEXINT

Use the CHAR2HEXINT function to inspect character data in its raw form as stored in the database:

```
select "uc" as "unicode", char2hexint("uc")
   from test_unicode_ddlorder by "uc";
```

```
unicode  Char2HexInt(uc)
Line1    004C  0069  006E  0065  0031
            L     i     n     e     1   🡐 Character by character mapping
Line2    004C00690006E00650032
Line3    004C00690006E00650033
```

Output 3. CHAR2HEXINT Function

```
CREATE TABLE grotto.test_latin_ddl
(
      "lc" VARCHAR(10) CHARACTER SET LATIN NOT CASESPECIFIC
) PRIMARY INDEX ( "lc" );
insert into test_latin_ddl ("lc")
   select "uc" from test_unicode_ddl;
select "lc" as "latin", char2hexint("lc")
   from test_latin_ddl order by "lc";
```

```
latin    Char2HexInt(lc)
Line1    4C696E6531
Line2    4C696E6532
Line3    4C696E6533
```

Output 4. CHAR2HEXINT Function Output for LATIN Characters

TRANSLATE

Use the TRNASLATE function to translate from LATIN to UNICODE or vice versa:

```
select "uc" as "unicode", char2hexint("uc"),
   translate("uc" using unicode_to_latin) as to_latin,
   char2hexint(to_latin)
from test_unicode_ddl order by "uc";
```

```
unicode   Char2HexInt(uc)         to_latin    Char2HexInt(to_latin)
Line1      004C0069006E00650031     Line1       4C696E6531
Line2      004C0069006E00650032     Line2       4C696E6532
Line3      004C0069006E00650033     Line3       4C696E6533
```

Output 5. Using TRANSLATE Function to Convert UNICODE to LATIN

The TRANSLATE function can also be used to test the validity of specific characters or strings.

The `U&'#XXXX#XXXX#XXXX'` notation illustrated in the code below is a way to define a Unicode character literal using hexadecimal notation:

```
select translate(_Unicode U&'#0031#0032#0033' uescape '#'
   using unicode_to_latin) as good_string;
```

```
good_string
123
```

Output 6. Using TRANSLATE Function to Check for Valid Characters in the Target Encoding

```
/* Unicode replacement character U+FFFD is invalid */
select translate(_Unicode U&'#FFFD' uescape '#'
   using unicode_to_latin) as bad_string;
```

```
*** Failure 5356 One of the characters or code point does not exist within
 the specified repertoire.
```

Output 7. Using TRANSLATE Function to Check for Valid Characters

CAUTION ABOUT MISUSING TERADATA LATIN

Some legacy Teradata implementations have stored multi-byte characters in Teradata LATIN (a single-byte encoding) without translation. Or they might have stored untranslated single-byte characters in Teradata LATIN. Using the default Teradata client character set (ASCII) allows data to stream in and out of the database without translation. These installations are treating character data as a byte stream instead of as a collection of properly translated strings.

This approach would sometimes save database space by not using Unicode. But the space savings are less of a concern with modern Teradata systems that support various compression features.

Working with untranslated data in Teradata can work for some operations, as long as the database just moves the character data around and does not need to know how it is encoded. But using LATIN to store untranslated data causes issues, including the following:

- Character column overflow or truncation
- Incorrect collation (sorting)

- Inability to use character literals in SQL text (for inserts, comparisons, and so on)

This practice is strongly discouraged. Character set LATIN should only be used to store data that can be accurately translated to Teradata LATIN. Data that cannot be translated to Teradata LATIN should be stored as Unicode in the database, and accessed using a client character set that matches the encoding of the data.

TERADATA CLIENT CHARACTER SETS

DEFAULT CLIENT CHARACTER SET

The default Teradata client character set is ASCII. ASCII does no translation or validation of character data as it moves between the client application and the database.

To protect from loading invalid data into Teradata, only use the default client character set ASCII for 7-bit ASCII (single-byte) data.

Always use a client character set that is appropriate for your data.

UTF-8 ENCODING

UTF-8 is a good choice for the client character set when working with international data. It can represent virtually all characters in all current world languages, while efficiently representing single-byte data. New applications and open-source projects are increasingly using UTF-8 as the recommended (or only) character encoding.

UTF-8 is one of the permanently enabled Teradata client character sets. No actions need to be taken by a Teradata administrator to make UTF-8 encoding available to applications.

UTF-8 is a variable length encoding. Table 2 shows how different numbers of bytes are used to represent ranges of characters.

In the first byte, if…	Then the character sequence is...
High-order bit is zero	one byte long. This leaves seven bits to encode information. If a character has a Unicode value that can be represented in seven bits, it is represented as a byte containing the Unicode value. For example, Unicode value 0x0041 is transformed to UTF-8 byte 0x41.
Three high-order bits are 110	two bytes long. The second byte has the two high order bits set to 10. There are five free bits in the first byte and six free bits in the second byte. This allows eleven bits to represent a numeric value. If a character has a Unicode value that can be represented in eleven bits, and cannot be represented by a shorter UTF-8 sequence, then it is represented as two bytes, where the free bits contain the Unicode value.
Four high-order bits are 1110	three bytes long. The second and third bytes have the two high order bits set to 10. There are four free bits in the first byte and six free bits in each of the second and third bytes. This allows sixteen bits to represent a numeric value. If a character has a Unicode value that can be represented in sixteen bits, and cannot be represented by a shorter UTF-8 sequence, then it is represented as three bytes, where the free bits contain the Unicode value. For example, Unicode value 0x3000 is transformed to UTF-8 byte sequence 0xE3 0x80 0x80.

Table 2. Variable Length Bit Patterns Used by the UTF8 Encoding Scheme

OTHER LANGUAGE-SPECIFIC TERADATA CLIENT CHARACTER SETS

Teradata comes with four permanently enabled client character sets. Other language-specific character sets can be installed and enabled by the Teradata database administrator.

1. Set the InstallFlag column to 'Y' for all character sets you want to activate:

```
UPDATE DBC.CharTranslationsV SET InstallFlag='Y'
WHERE CharSetId = <charset_id>;
```

   or:

```
UPDATE DBC.CharTranslationsV SET InstallFlag='Y'
WHERE CharSetName = '<charset_name>';
```

2. Perform a full restart of Teradata Database using the tpareset utility.

The limit is 16 active character sets. For more details see the Teradata International Character Support manual.

Table 3 lists some common language-specific Teradata client character sets with their corresponding SAS session encodings. Not all available encodings are listed here.

Character Set Type	Name	SAS Encoding (UNIX)	SAS Encoding (Windows)
Permanently Enabled Character Sets	ASCII (No translation by client)		
	EBCDIC (IBM Mainframe)		
	UTF8 (variable width characters)	utf-8	utf-8
	UTF16 (double-byte characters)		
Japanese Character Sets	KATAKANAEBCDIC		
	KANJIEBCDIC5035_0I		
	KANJISJIS_0S	shift-jis	shift-jis
	KANJIEBCDIC5026_0I		
	KANJIEUC_0U	euc-jp, dec-jp	euc-jp, dec-jp
	KANJI932_1S0		ms-932
Chinese Character Sets	SCHEBCDIC935_2IJ		
	SCHGB2312_1T0		
	TCHEBCDIC937_3IB		
	SCHGB2312_1T0	euc-cn, dec-cn	
	TCHBIG5_1R0	euc-tw, dec-tw, big5	euc-tw, dec-tw, big5
	SCHINESE936_6R0		euc-cn, dec-cn, ms-936
	TCHINESE950_8R0		ms-950
Korean Character Sets	HANGULEBCDIC933_1II		
	HANGULKSC5601_2R4	euc-kr	
	HANGUL949_7R0		euc-kr, ms-949
Single-Byte International Character Sets	THAI874_4A0 (Thai)	thai	thai
	LATIN1250_1A0 (Czech, Croatian, Albanian, Hungarian, Polish, Romanian, Serbian)		wlatin2

Character Set Type	Name	SAS Encoding (UNIX)	SAS Encoding (Windows)
	CYRILLIC1251_2A0 (Cyrillic and Russian)		wcyrillic
	HEBREW1255_5A0		whebrew
	ARABIC1256_6A0		warabic
	LATIN1254_7A0 (Turkish)	latin5	latin5, wturkish
	LATIN1258_8A0 (Vietnamese)		wvietnamese
	LATIN1252_3A0 (English, Western European) LATIN1252_0A	latin1	wlatin1

Table 3. Teradata Client Character Sets with Corresponding SAS Encodings

SPECIFYING THE CLIENT CHARACTER SET

The Teradata session encoding that is used for a SAS session is set in the Teradata client (TTU) configuration file clispb.dat. To change the client character set add two entries to this file:

1. charset_type, which tells TTU how to interpret the character set ID (N=by name, C=by code value)
2. charset_id, which is the character set's name (charset_type=N) or its numeric ID (charset_type=C)

```
# cat clispb.dat
req_buf_len=1024
resp_buf_len=8192
.
.
.
charset_type=N
charset_id=utf8
```

Use caution when editing the clispb.dat file. The last line of this file must end with a carriage return on Windows systems, and this file cannot contain a blank line at the end.

For All Sessions on the SAS Server

To set a new client character set for all SAS sessions from a specific server, modify the clispb.dat file in its installed location (the path might vary depending on the TTU version):

```
/opt/teradata/client/15.10/etc/clispb.dat
```

For a Specific SAS Session

To create a personal client configuration file, copy clispb.dat from its installed location to a separate directory and modify the private copy.

At run time, set the COPLIB environment variable to point to the directory containing the custom file:

```
# export COPLIB=/myapp/home/COPLIB
# grep charset $COPLIB/clispb.dat
charset_type=N
charset_id=utf8
```

CHECKING THE CLIENT CHARACTER SET FROM SAS

To confirm that connections to Teradata that are made by SAS/ACCESS are using the UTF-8 client character set, use PROC SQL to submit the HELP SESSION statement:

```
proc sql;
connect to teradata (server=tdbms user=grotto password=******);
```

```
 select Character_Set from connection to teradata (
     help session
 );
disconnect from teradata;
quit;
```

```
Character_Set
UTF8
```

Output 8. Teradata Client Session Encoding Displayed from a SAS Session

UNICODE PASS THROUGH

Teradata database version 16.00 and the corresponding TTU 16.00 include a new feature called Unicode Pass Through (UPT). UPT is a Unicode error handling feature that allows the user to import and export additional characters that would normally be considered invalid. This expands the characters that are available to Unicode users so that they can store, retrieve, and analyze emojis and other ideographs.

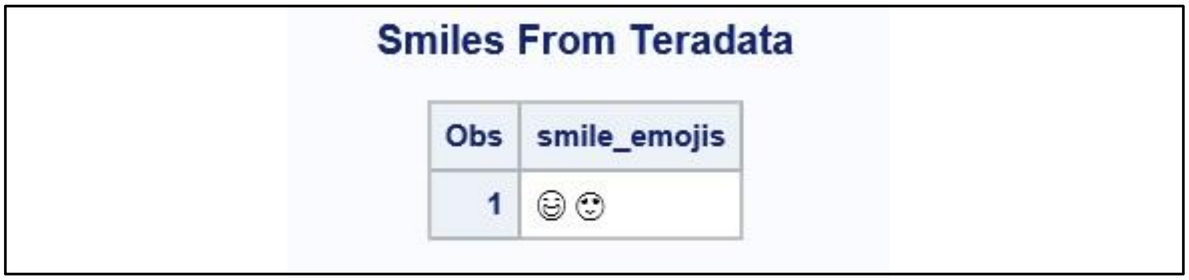
To use the UPT feature, the Teradata client character set and SAS encoding should both be set to UTF-8.

```
libname s '.';
data s.smiles;
   length smile_emojis $20;
   smile_emojis = 'f09f98842020f09f988d'x;  output;  /* smile <sp><sp> smile */
run;
proc contents data=s.smiles;  run;  /* confirms utf-8 encoding */

ods html file='smiles_out1.html';   /* display SAS data set as HTML */
title 'Smiles From SAS';
proc print data=s.smiles;  run;
ods html close;

libname td teradata server=td16 user=grotto pwd=******
    DBCONINIT='SET SESSION CHARACTER SET UNICODE PASS THROUGH ON;';
proc delete data=td.smiles; run;
data td.smiles(dbtype=(smile_emojis='VARCHAR(100) CHARACTER SET UNICODE'));
   set s.smiles;
run;

ods html file='smiles_out2.html';   /* display Teradata table as HTML */
title 'Smiles From Teradata';
proc print data=td.smiles;  run;
ods html close;
```


Smiles From Teradata

Obs	smile_emojis
1	😄😍

Figure 2. HTML Rendering of Emojis Loaded from SAS into Teradata

For more information about the UPT feature see the Teradata International Character Set Support manual for TD 16.00.

SAS ENCODING CONFIGURATION

SAS FOUNDATION SESSION AND SAS/ACCESS INTERFACE TO TERADATA

These examples launch SAS using the US-English locale. Your locale might vary, depending on the requirements at your site.

SAS on Windows

To launch SAS with UTF-8 encoding on Windows, use the supplied UTF-8 configuration file. If a custom TTU configuration file has been created, then include the COPLIB environment variable to identify the directory that contains the modified clispb.dat file:

```
"C:\Program Files\SASHome\SASFoundation\9.4\sas.exe"
-config "C:\Program Files\SASHome\SASFoundation\9.4\nls\u8\sasv9.cfg"
-set COPLIB "C:\myapp\home\COPLIB" %*
```

SAS on Linux or UNIX

To launch SAS with UTF-8 encoding on Linux or UNIX, use the supplied UTF-8 start script. If a custom TTU configuration file has been created, then include the COPLIB environment variable to identify the directory that contains the modified clispb.dat file:

```
/sashome/SASFoundation/9.4/bin/sas_u8 -encoding "UTF-8" -locale "en_us"
-set COPLIB "/myapp/home/COPLIB" $*
```

VERIFYING THE SAS SESSION ENCODING

Use the OPTIONS procedure to confirm that SAS is running with UTF-8 encoding and the desired locale:

```
proc options group=languagecontrol; run;
```

```
Group=LANGUAGECONTROL
 DATESTYLE=MDY     Specifies the sequence of month, day, and year when
ANYDTDTE, ANYDTDTM, or ANYDTTME informat data is ambiguous.
 DFLANG=ENGLISH    Specifies the language for international date informats
and formats.
 .
.
.
 ENCODING=UTF-8    Specifies the default character-set encoding for the SAS
session.
 LOCALE=EN_US      Specifies a set of attributes in a SAS session that
reflect the language, local conventions, and culture for a
                   geographical region.
 NONLSCOMPATMODE   Encodes data using the SAS session encoding.TF8
```

Output 9. SAS Session Language Control Options Including Encoding

SAS EMBEDDED PROCESS FOR TERADATA

The SAS Embedded Process is a SAS processing engine that runs on Teradata, in parallel, alongside the database. The SAS Embedded Process includes a SAS DS2 run-time package and a set of Teradata table functions, views, and Teradata stored procedures that implement the DBS-to-embedded process interface.

For more information about how the SAS Embedded Process and Teradata interact for scalable in-database computation, refer to the paper "Parallel Data Preparation with the DS2 Programming Language" (Secosky, Ray, Otto).

International data must be stored in Teradata UNICODE encoding when it is used with the SAS Embedded Process for parallel in-database applications, including the SAS Scoring Accelerator for Teradata, SAS In-Database Code Accelerator for Teradata, and SAS Data Quality Accelerator for Teradata.

When the SAS Embedded Process is called from SAS, the Teradata SQL that activates the embedded process needs to set the ENCODING= parameter to UNICODE and the LOCALE= parameter to signal the embedded process that it is receiving Unicode data from the database. This parameter engages the appropriate transcoding routines inside the embedded process to translate character data into the encoding and locale that matches the SAS environment.

Figure 3 illustrates the interactions between SAS and Teradata when the SAS Embedded Process processes international data. Once again, we assume that the user is working with the Turkish language in a SAS session with ENCODING=latin5.

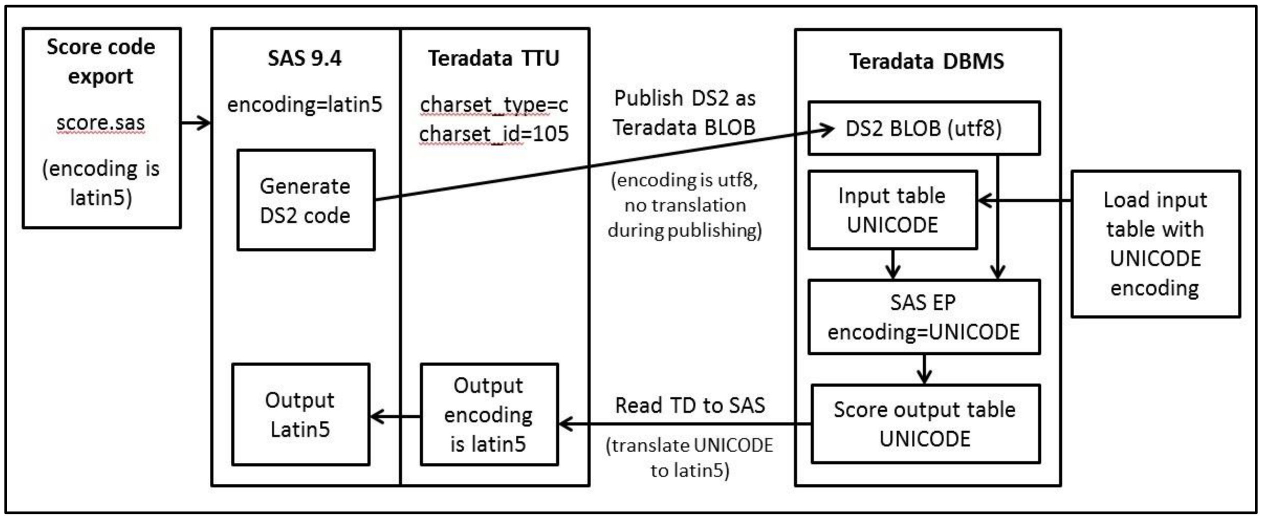


Figure 3. In-Database Processing with the SAS EP Requires UNICODE Input Data

SAS In-Database Code Accelerator for Teradata and the parallel data feeders for SAS High-Performance Analytics and SAS Cloud Analytic Services (CAS) automatically insert the embedded process encoding parameter in the SQL that they generate to activate the embedded process.

For applications that activate the SAS Embedded Process via a stored procedure interface, the user must insert the ENCODING parameter, as shown in this example:

```
proc sql;
  connect to Teradata (
     server=tdbms user=grotto password=****** mode=Teradata
  );
  execute (
     call SAS_SYSFNLIB.SAS_SCORE_EP
     (
        'INQUERY=select seq, CUST_IDV_EMP_TP from "grotto"."t_latin5_in"',
        'MODELTABLE="grotto"."sas_model_table"',
        'MODELNAME=latin5_model',
        'OUTTABLE="grotto"."t_latin5_out"',
        'OUTKEY=seq',
        'OPTIONS=ENCODING=UNICODE;LOCALE=en_us;JOURNALTABLE=t_latin5_jnl;'
     )
```

```
    ) by teradata;
    disconnect from teradata;
  quit;
```

SAS THREADED KERNEL ACCESS DRIVER FOR TERADATA

The SAS threaded kernel (TK) SAS/ACCESS Driver is used to connect from the FEDSQL procedure to Teradata. The TK driver accepts a CLIENT_ENCODING connection option that sets the client session encoding to the specified character set, regardless of the character set values in clispb.dat file:

```
PROC FEDSQL nolibs noerrorstop noprompt=
   "(CATALOG=TERADATA;DRIVER=TERADATA;SERVER=tdbms;UID=grotto;PWD=******;
     CLIENT_ENCODING=UTF8;)";
```

SAS DATA CONNECTOR AND DATA CONNECT ACCELERATOR FOR TERADATA

The SAS Data Connector to Teradata and the SAS Data Connect Accelerator for Teradata are the primary tools to access Teradata data using the CAS engine within the SAS Viya architecture.

UTF-8 is the only SAS session encoding that is supported by SAS Viya. Ensure that the character set options in the clispb.dat file are set as follows:

```
charset_type=N
charset_id=utf8
```

To tell where to find the Teradata client configuration file, modify the cas.setting file on the CAS controller node in the directory:

```
SASHOME/SASFoundation
```

Add the following lines:

```
export COPERR=Teradata-TTU-install-path/lib
export COPLIB=Directory-that-contains-clispb.dat
export NLSPATH=Teradata-TTU-install-path-msg-directory:$NLSPATH
export LD_LIBRARY_PATH=Teradata-TTU-install-path-lib64-
directory:$LD_LIBRARY_PATH
```

CONCLUSION

Working with international data is often shrouded in a fog of mystery and confusion. The fog will be lifted when you carefully consider the path that your data takes as it moves between systems, configuring each component along the way so that they all agree about the encoding. Remember to consider all of the data handlers along the journey:

1. Teradata server character set
2. Teradata client character set
3. SAS encoding

REFERENCES

- *SAS/ACCESS For Relational Databases: Reference, Ninth Edition.* Available at

  http://support.sas.com/documentation/cdl/en/acreldb/69580/HTML/default/viewer.htm#titlepage.htm

- *SAS Manuals to configure SAS with Teradata TPT - SAS Companion Manual For Windows.* Available at

  http://support.sas.com/documentation/installcenter/en/ikfdtnwx6cg/66385/PDF/default/config.pdf

- *SAS 9.4 Companion for UNIX Environments, Sixth Edition.* Available at http://support.sas.com/documentation/cdl/en/hostunx/69602/PDF/default/hostunx.pdf

- *Multilingual Computing with SAS® 9.4.* Available at https://support.sas.com/resources/papers/Multilingual_Computing_with_SAS_94.pdf
- *SAS 9.4 National Language Support (NLS): Reference Guide, Fifth Edition*. Available at http://support.sas.com/documentation/cdl/en/nlsref/69741/HTML/default/viewer.htm#titlepage.htm
- *SAS® Viya™ 3.1: Deployment Guide.* Available at http://go.documentation.sas.com/api/docsets/dplyml0phy0lax/1.1/content/dplyml0phy0lax.pdf
- *Teradata Tools and Utilities – Windows Installation Package.* Available at https://downloads.teradata.com/download/tools/teradata-tools-and-utilities-windows-installation-package
- *Teradata International Character Set Support (15.00/15.10).* Available at http://www.info.teradata.com/download.cfm?ItemID=1001863
- *Teradata Call-Level Interface: Version 2 Reference for Workstation-Attached Systems (15.10).* Available at http://www.info.teradata.com/HTMLPubs/DB_TTU_15_10/Interface_Tools/B035_2418_035K/2418title.html
- *Teradata Database: SQL Data Types and Literals (15.10).* Available at http://www.info.teradata.com/download.cfm?ItemID=1003561
- *UNICODE Server Character Set (B035-1056-111K)*. Available at http://www.info.teradata.com/download.cfm?ItemID=1003561
- *SAS329-2014 Parallel Data Preparation with the DS2 Programming Language*. Available at http://support.sas.com/resources/papers/proceedings14/SAS329-2014.pdf

ACKNOWLEDGMENTS

The authors extend their heartfelt thanks and gratitude to these individuals for their contributions and assistance with reviewing this paper:

Meredyth Bass, SAS Institute Inc., Cary, NC

Tatyana Petrova, SAS Institute Inc., Cary, NC

Junichi Arai, SAS Institute Inc., Cary, NC

Heather Burnette, Teradata Corporation, Dayton, OH

Ali Pasha, Teradata Corporation, Dayton, OH

CONTACT INFORMATION

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

Salman Maher
SAS Institute, Inc.
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Salman.Maher@sas.com
http://www.sas.com

Greg Otto

Teradata Corporation
100 SAS Campus Drive
Cary, NC 27513
Greg.Otto@teradata.com

Austin Swift
SAS Institute, Inc.
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Austin.Swift@sas.com
http://www.sas.com