Technical Paper

# Multilingual Computing with SAS® 9.4

Content provider for Multilingual Computing with SAS® 9.4 was Manfred Kiefer, Principal Domain Specialist in the Software Globalization Division.

## Table of Contents

## Introduction

Multilingual data, that is, data coming from a variety of regions, should usually be stored and processed in Unicode. SAS supports Unicode as session encoding in the form of UTF-8. This is the ideal solution for making your SAS applications useable in many languages.  To a certain extent, you can use multiple languages with other encodings as long as those languages use a compatible encoding, that is an encoding that supports all those languages. This works for example with Windows or UNIX Latin1 that have support for many Western European languages.  However, z/OS using EBCDIC with its huge variety of code pages is not well suited for handling multilingual data because even Western European languages use several incompatible code pages.

Regarding the use of Unicode Server, you need to evaluate the pros (such as one single repository for all possible locales, or a seamless handling of data across all platforms) and cons (such as increased storage space, or special functions needed for data manipulation). You might come to the conclusion that a legacy encoding would suffice for processing your data. If not, Unicode Server is the right choice. How to work with multilingual data in a legacy environment is also explained farther down in the text.

## Installing and Configuring the SAS System

In SAS 9.4, you can:

- select the language for the SAS Deployment Wizard and SAS products[1]
- select Regional Settings that apply to the SASFoundation locale and the language of SAS metadata
- set the locale to the correct value for SASFoundation and Java rich clients if it was set incorrectly
- set your Java rich clients to run an English User Interface with a non-English SAS server.

Beginning with 9.4, the ENCODING system option is set explicitly in the configuration file for single-byte languages and Unicode support. It is not set for double-byte languages.
The SAS Deployment Wizard allows the user to decide which set of languages to install for the SAS products that are listed.
Three images are automatically deployed for SAS on all Windows and UNIX machines:

- 'English' is a single-byte SAS image that displays an English User Interface and English messages by default. The LOCALE and ENCODING options for the English image are set to match the Regional settings or, if the Regional Settings selection is an Asian language, it sets LOCALE and ENCODING to support en_US.
- 'English with DBCS' is a double-byte SAS image that displays English User Interface and English SAS messages by default. This image supports languages that require a double-byte character set, such as Chinese. If a double-byte language is selected later in the Regional Settings dialog, the LOCALE option in the 'English with DBCS support' config file is set to match. Otherwise, the LOCALE defaults to ja_JP.
- 'Unicode Support' is installed for all Windows and UNIX deployments, even if the SAS server is not configured for Unicode support. The ENCODING option is set to utf-8. The LOCALE of the Unicode server is set to match the Regional Settings locale selection.

The SAS Deployment Wizard (SDW) prompts the user for the available localizations to install.
If you have chosen to install any of the localized SAS images, you will find a folder under !SASROOT\nls for each. The folders are named with a SAS specific , two character language code; for example 'fr' for French. Each contains its own configuration file with a LOCALE option value set to match the localization. For example, the configuration file for French (fr) has a LOCALE value with a French language code.

On Windows for each localization that is installed, a nls\xx sasv9.cfg file is created where 'xx' represents the SAS specific, two character language code . In addition to creating a config file for each installed localization, a config file is also created for English with DBCS as well as UTF-8 (u8).

On UNIX, start-up scripts files are created for all installed languages. The default "sas" start-up script has a link to the

[1] The selection might fail if an Asian language (e.g. Chinese or Japanese) was used as SDW interface language when running SDW on an English OS. This will be fixed in a later release of SAS.

default language. In order to change the default language, the user simply changes this link to point to the appropriate language script. For example, sas_en invokes the English version of SAS 9.4 Foundation. SAS 9.4 Foundation also creates a separate configuration file for each language installed (including English). These language-specific configuration files are !SASROOT/nls/<lang>/sasv9.cfg for each respective language. An additional configuration file that is language independent is !SASROOT/sasv9.cfg. This master configuration file in !SASROOT is used by all languages in addition to the language-specific files in !SASROOT/nls/<lang>/.

You need to check whether the fonts for the languages that you need (e.g. Chinese or Korean) are available in the /usr/share/fonts directory. If not, then you need to install those fonts. Usually, there are GUI tools to accomplish this, for both the host side and the X server. Several online resources exist on explaining how to install and use TrueType fonts with the X Window system**.**

To allow SAS Java client applications, such as SAS® Data Integration Studio, to display Unicode characters that you may have to modify the font configuration file(s) that the Java Runtime Environment (JRE) uses. The SAS provided JRE should have the mapping to the SAS fonts and it should be able to display all characters correctly. However, a Vendor supplied JRE might not have valid mappings for some locales. Therefore, as long as using a vendor JRE, you might have to modify it.

JRE uses font configuration files to map logical font names to physical fonts. For further reference and documentation, see Java SE Documentation. To support Unicode display, 'jre/lib/font.properties' might have to be modified and the languages' fonts need to be added to each section. There are different properties files based on the platform. An example is shown in the appendix.

On z/OS media is available in several encoded versions, which support different locales and regions. The **encoding used by SAS must be compatible with the encoding of the installed media**. If an incompatible encoding is set, then the following warning is displayed in the SAS log:

WARNING: There is an incompatibility between session encoding and the SASHELP encoding. When such a mismatch occurs, some features might not behave as expected. For additional information, contact your Site Administrator.

NOTE:  SAS on z/OS **does not support UTF-8** as session encoding.

A dialog box in the SAS Deployment Wizard allows the installer to select the locale to use for SAS software being installed on your system. The selected locale is set as the value of the SAS LOCALE system option in the configuration file used to initialize the SAS System.

Beginning in 9.4, the ENCODING system option is set explicitly in the configuration file for all images except the DBW0 image.

The path encoding, set in the TKMVSENV file, must also match the explicit or implicit ENCODING option, or must be set to the "compiler" encoding, open_ed-1047. The locale and encodings are specified in the TKMVSENV file that sets the environment variables for the SAS environment.

For example, if you have selected German_Germany from the locales selector for a W3 installation, then the configuration file and TKMVSENV file is updated accordingly to your selection:

```
CONFIG(ENW3):
ENCODING=open_ed-1141
LOCALE=de_DE  TKMVSENV(TKMVENW3):
set TKOPT_ENV_ENCODING=open_ed-1141
set TKOPT_ENV_ENCODING_PATH=open_ed-1141
set TKOPT_ENV_LOCALE=de_DE
```

## New TrueType fonts

For 9.4, the font licensing agreement with Monotype Imaging has been updated to include two new fonts: Arial Unicode MS and Times New Roman Uni. These two new fonts replace the eight Unicode fonts Monotype Sans WT (J,K,SC,TC) and Thorndale Duospace WT (J,K,SC,TC).

| Font Name | Languages Supported | Font Description |
|---|---|---|
| Arial Unicode MS | Arabic, Armenian, Basic Latin, Bengali, Bopomofo,Cyrillic, Devanagari, Georgian, Greek and Coptic, Gujarati,Gurmukhi, Hangul Jamo, Hebrew, Hiragana, Kanbun, Kannada,Katakana, Lao, Malayalam, Oriya, Tamil, Telugu,Oriya, Tamil, Telugu, Thai,Tibetan. | sans-serif |
| Times New Roman Uni | Arabic, Basic Latin, Bopomofo, Cyrillic,Devanagari,Georgian, Greek and Coptic,Gujarati, Hangul Jamo,Hebrew, Hiragana, Kanbun, Katakana, Lao, Mongolian,Tamil, Telugu, Thai, Tibetan. | serif |

*Table 1: Multilingual Unicode TrueType Fonts*

For Asian languages the Sim Hei, SimSun, and NSimSun fonts are replaced by CSongGB18030C-Light, CSongGB18030C-LightHWL, MYingHei_18030_C-Medium, and MYingHei_18030_C-MediumHWL.

| Language Supported | Font Name | Character Set |
|---|---|---|
| Japanese | MS Gothic, MS UI Gothic, MS PGothic | Shift JIS |
| | MS Mincho, MS PMincho | Shift JIS |
| Korean | Gulim, GulimChe, Dotum, DotumChe | KSC5601 |
| | Batang, BatangChe, Gungsuh, GungsuhChe | KSC5601 |
| Simplified Chinese | CSongGB18030C-Light<br>CSongGB18030C-LightHWL<br>MYingHei_18030_C-Medium<br>MYingHei_18030_C-MediumHWL | GB18030 and GB2312 |
| Traditional Chinese | HeiT | Big5 |
| | MingLiU, MingLiU_HKSCS, PMingLiU | Big5 |

*Table 2: Asian Monolingual TrueType Fonts*

The fonts that are supplied by SAS and the fonts that are already installed on Windows are automatically registered in the SAS registry when you install SAS. Fonts already installed on UNIX and z/OS must be registered manually in the SAS registry after you install SAS.

## Language of Metadata

The language of the locale selected in the SDW for SAS products is used as the language of metadata properties set during deployment. The locale data contains a primary language and, for some locales, it also contain a secondary language. (See the example below.)
If the properties are not available for the primary language, metadata uses the secondary language, if available. If no secondary language is provided for the locale or the data for the secondary language is not available, English is used. For example, the primary SAS 2-byte language code for French (Canada) is 'fc' and the secondary language is 'fr'. The SDW tries to use 'fc' properties first. If not available, it uses 'fr'. Finally, if those are not available it uses 'en'.

## Change the locale of a deployment

The Locale Setup Manager (LSM) was provided in early SAS9 to allow customers to change the locale for Java vertical products on Windows. The Locale Setup Manager for 9.4 has been moved into the SAS Deployment Manager (SDM) and is available customers on Windows and UNIX.
The enhanced SAS Deployment Manager addresses two specific requirements:

- Users who choose the wrong locale during deployment can set the correct one in their SASFoundation configuration and all Java rich clients installed on the machine.
- Users who allowed the locale to default to the OS setting, but want the client User Interface to be English can use SAS Deployment Manager to set this right.

They just need to select all products listed and choose the new locale. What happens when a new locale is selected for all products:

- The Deployment registry is updated.
- The SASFoundation config file is updated to use the selected locale. Java rich client ini files are updated with language and country settings that match the locale.

## Working with Multi-byte Data

The SAS System provides many string functions and call routines that can be used to manipulate characters and strings. The original SAS string-handling functions assume the size of a character is always one byte, which is true with a single-byte encoding. However, this assumption is not correct with a double-byte or multi-byte encoding. For example, UTF-8 is a variable-width multi-byte encoding. One character in the UTF-8 encoding can be 1 byte, 2 bytes, 3 bytes, or even 4 bytes. Using the original SAS string-handling functions with double or multi-byte data can lead to unexpected behavior, such as data truncation.
To resolve issues these string functions can cause for multi-byte data, SAS provides a set of string functions, called K functions, which do not make assumptions about the size of a character in a string. To use K functions, you need to understand the difference between byte-based offset and character-based offset.
A *byte-based offset assumes that the starting position specified for a character is the byte position of that character in the string. For single-byte data, since one character is always one byte in length, you can assume that the second character in the string begins in byte two of the string. However, if the data in the string is multi-byte, the data in the second byte can be one of the following, depending on the data and encoding of the data:*

- the second character in the string,
- the second byte of a 2-byte character,
- or the first byte of the first multi-byte character in the string.

A *byte-based length* represents the number of bytes in the string.
A *character-based offset* assumes that the position specified is the position of the character in the string. For all encodings, a character-based position of 2 is always the second character in the string. You cannot assume that you know the size of the characters in the string. A *character-based length* represents the number of characters in the string.
K functions use a character-based offset or length, which does not take into consideration the byte position of the character in the string. K functions can be used for processing single-byte and multi-byte data in SAS. A complete list of K functions and documentation on their use can be found in the SAS 9.4 National Language Support (NLS): Reference

Guide. There are some compatibility issues to consider when you use K functions. For example, while the original TRIM function does correctly process multi-byte characters, it does not remove double-byte blanks so that there are some behavior differences. For details about these differences refer to “Internationalization Compatibility for SAS String Functions” in the SAS 9.4 National Language Support (NLS): Reference Guide.
Before replacing all of the original SAS string-handling functions with K functions, however, examine your SAS code. If the string function processes data that only contains single-byte characters, there is no need to use K functions. For example, strings containing XML tags do not require the use of K functions. Knowing the character data that in your SAS programs and how it is processed can save unnecessary updates to your SAS code.

## Reading and Writing Data

Data can be read from and written to:
- External files
- SAS libraries
- DBMS tables.

In each case, it is important to know about the particular encoding of the source. Let us take a closer look at how the encoding of source data is determined.

### External Files

SAS reads and writes external files using the current session encoding. This means that the system assumes that the external file uses the same encoding as the SAS session. An external file can contain only character data or a mixture of character and binary data. In either case, the encoding for the character data in the external file can be different from the SAS session encoding.
When a file contains only character data and its encoding is different from the SAS session encoding, use the ENCODING= option on the FILENAME, INFILE, or FILE statement to tell SAS what the file encoding is so that SAS can transcode the data from its original encoding to the current SAS session encoding; for example:

```
filename extfile 'external-file' encoding=wlatin1;
data contacts;
infile extfile;
length name $ 20 first $ 20;
input name first;
run;
```

See the SAS 9.4 National Language Support Reference Guide for details about using the ENCODING= option. When an external file contains a mix of character and binary data, you must use the KCVT function or the KPROPDATA function to convert individual fields from the file encoding to the session encoding. See the SAS 9.4 National Language Support Reference Guide for details about using the KCVT or KPROPDATA functions.

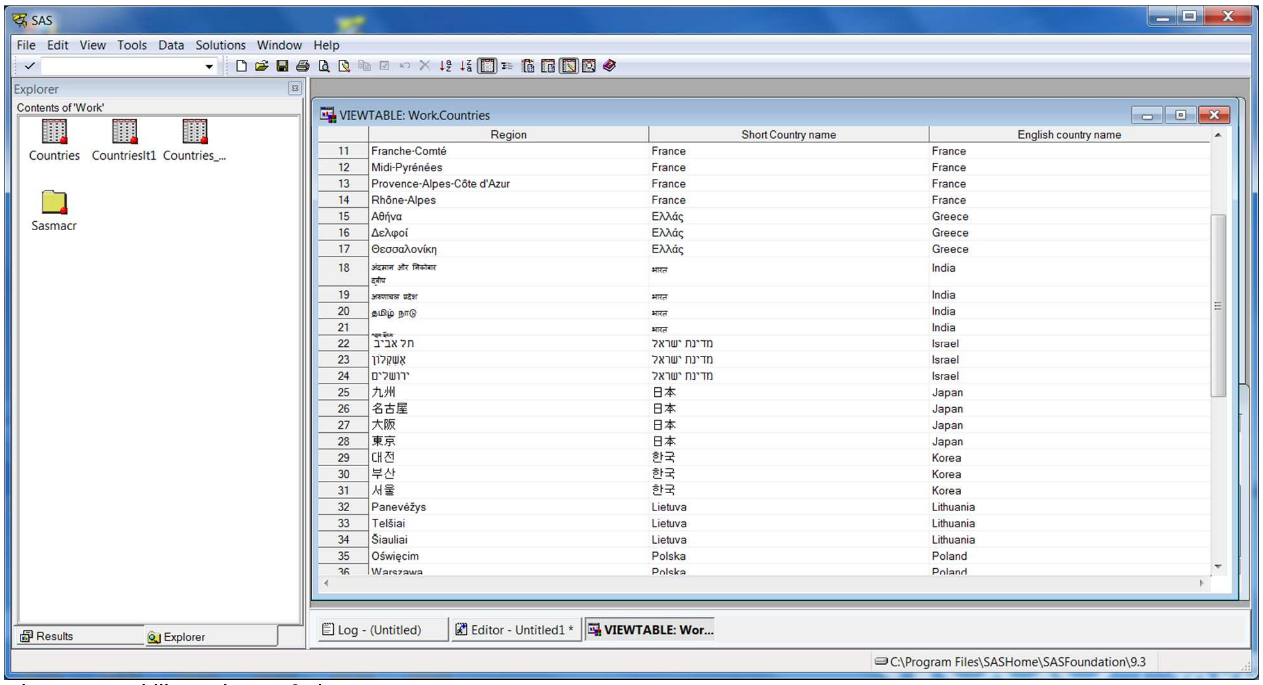
### SAS Libraries

SAS 9 data sets have an ENCODING attribute. When the data set encoding is different from the session encoding, the cross-environment data access (CEDA) facility automatically transcodes character data when it is read and when it is saved. Data sets from earlier releases do not have an encoding attribute, so you must specify the encoding of the incoming data with the ENCODING= DATA step option or the INENCODING= LIBNAME option. Note: If there is no encoding information in the data set, the session encoding is assumed.

If your session encoding is UTF-8, you need to increase variable lengths to prevent truncation during transcoding of the data to UTF-8. UTF-8 characters can require up to 4 bytes of memory. The length of your variables depends on the characters that are contained in your data. If your data consists of Asian characters, you  need to consider multiplying the number of characters by 4. However, for other data, 2.5 is a reasonable expansion.  The CVP engine provides an easy way to convert your files and avoid truncation problems. CVP is an engine that must be specified in the LIBNAME statement on the input data, so truncation does not occur during transcoding. By specifying the CVP engine, the default expansion is 1.5 times the current variable length. There are options to specifically set the expansion. See SAS Help for information about CVPMultiplier, CVPBytes, and CVPEngine.

Generally, it makes sense to read in or write to data only with an encoding that is compatible to the SAS session encoding - otherwise you get transcoding errors. In some cases, it can be appropriate to bypass transcoding errors so that users can finish their SAS jobs. In SAS 9.4 the new object spawner option 'ignoretranscodeerror' allows IOM servers to ignore transcoding failures. However, this option should be used under the direction of Tech Support in cases where customers are unable to correct transcoding issues with their data.

As mentioned, it does not make sense to transcode characters that are not representable in a particular encoding. If you know your data, you can subset them by encoding and transcode accordingly.  If there is no “language tag” in the data that would allow creating subsets per language or language group, any character that is not compatible with the target encoding is replaced by the default replacement character, which is 0x1a for ASCII-based encodings[2]. For example, if you have a UTF-8 data set with French, Greek, Hindi, and other characters in it, and you want to transcode those to WLATIN1, this would work for characters from French and other Western European languages. For all other characters that you run into transcoding errors and the code  fails.



VIEWTABLE: Work.Countries

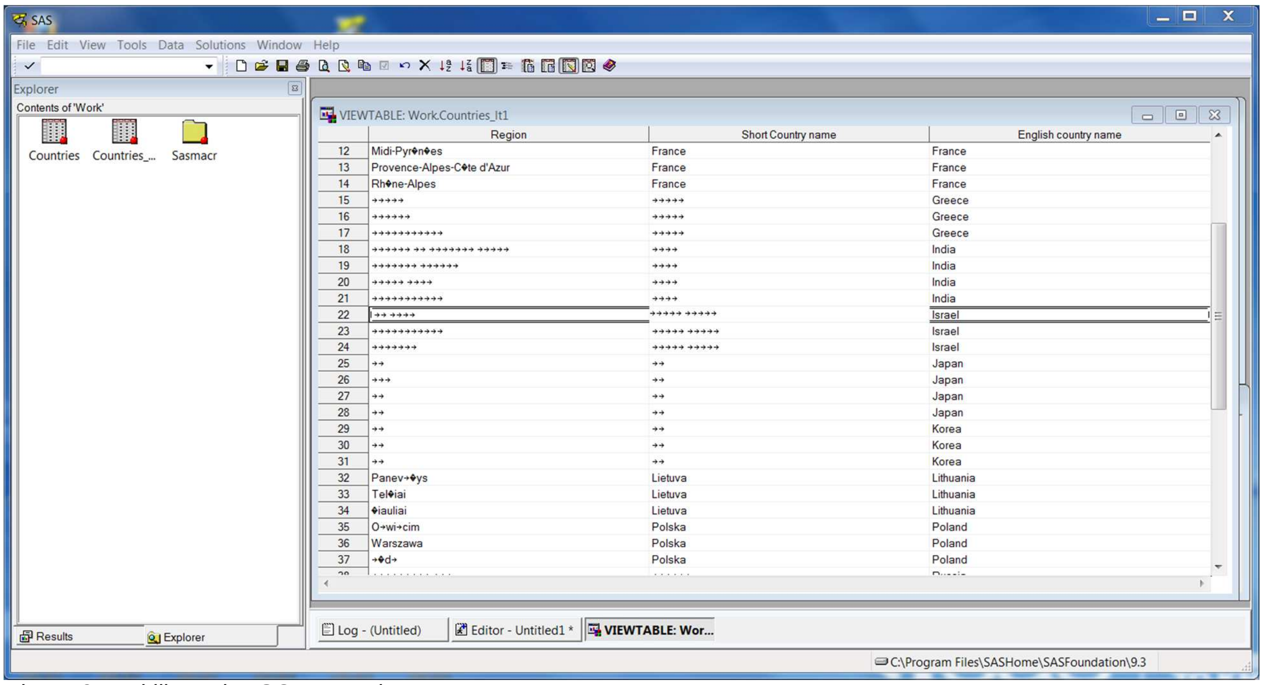
| | Region | Short Country name | English country name |
|---|---|---|---|
| 11 | Franche-Comté | France | France |
| 12 | Midi-Pyrénées | France | France |
| 13 | Provence-Alpes-Côte d'Azur | France | France |
| 14 | Rhône-Alpes | France | France |
| 15 | Αθήνα | Ελλάς | Greece |
| 16 | Δελφοί | Ελλάς | Greece |
| 17 | Θεσσαλονίκη | Ελλάς | Greece |
| 18 | अंदमान और निकोबार द्वीप | भारत | India |
| 19 | अरुणाचल प्रदेश | भारत | India |
| 20 | தமிழ் நாடு | भारत | India |
| 21 | [illegible] | भारत | India |
| 22 | תל אביב | מדינת ישראל | Israel |
| 23 | אַשְׁקְלוֹן | מדינת ישראל | Israel |
| 24 | ירושלים | מדינת ישראל | Israel |
| 25 | 九州 | 日本 | Japan |
| 26 | 名古屋 | 日本 | Japan |
| 27 | 大阪 | 日本 | Japan |
| 28 | 東京 | 日本 | Japan |
| 29 | 대전 | 한국 | Korea |
| 30 | 부산 | 한국 | Korea |
| 31 | 서울 | 한국 | Korea |
| 32 | Panevėžys | Lietuva | Lithuania |
| 33 | Telšiai | Lietuva | Lithuania |
| 34 | Šiauliai | Lietuva | Lithuania |
| 35 | Oświęcim | Polska | Poland |
| 36 | Warszawa | Polska | Poland |

*Figure 1: Multilingual UTF-8 data set*

```
ERROR: Some character data was lost during transcoding in the data set WORK.COUNTRIES_LT1. 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.
```

To avoid the error, you might tag the output data file with an encoding attribute of “ASCIIANY”.  This works, however, all Greek and Hindi characters, for example, is replaced by default replacement characters.  And the conversion does not even work for Western European characters:

[2] For EBCDIC the substitution character is 0x3F; UTF-8 uses 0xEFBFBD, UTF-16 0xFFFD, and UTF-32 0x0000FFFD.

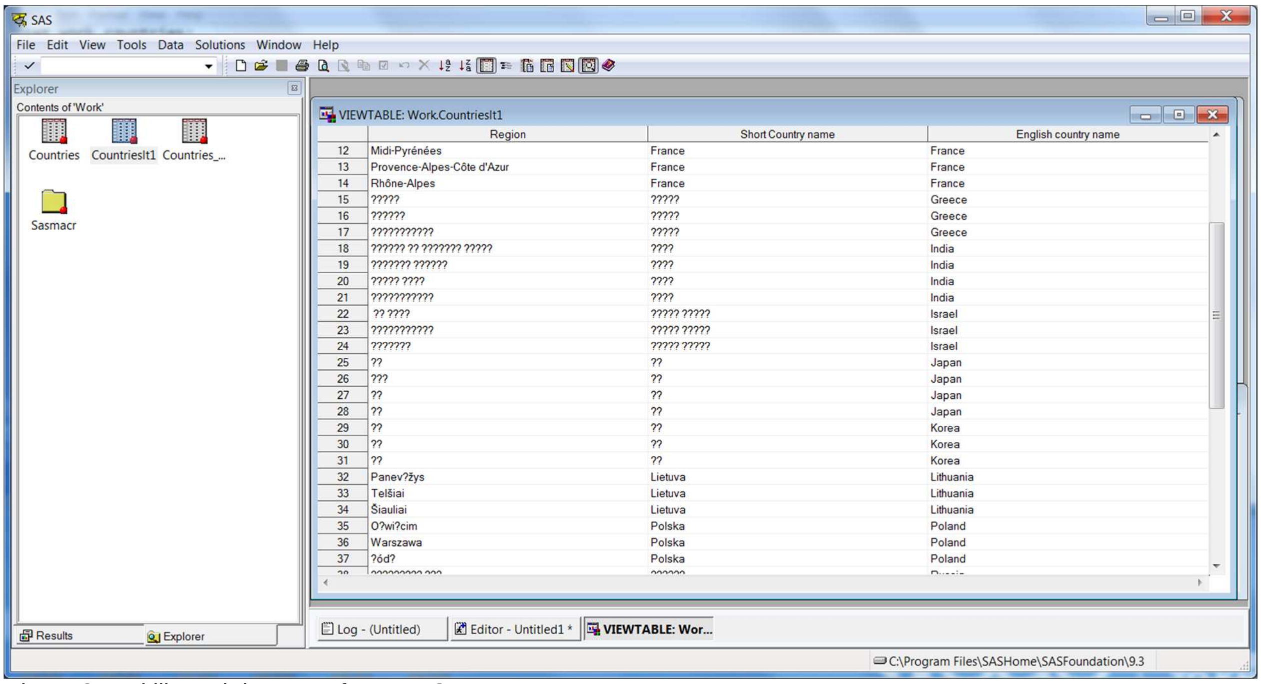

VIEWTABLE: Work.Countries_lt1

| | Region | Short Country name | English country name |
|---|---|---|---|
| 12 | Midi-Pyr◆n◆es | France | France |
| 13 | Provence-Alpes-C◆te d'Azur | France | France |
| 14 | Rh◆ne-Alpes | France | France |
| 15 | →→→→→ | →→→→→ | Greece |
| 16 | →→→→→→ | →→→→→ | Greece |
| 17 | →→→→→→→→→→→ | →→→→→ | Greece |
| 18 | →→→→→→ →→ →→→→→→→ →→→→→ | →→→→ | India |
| 19 | →→→→→→→ →→→→→→ | →→→→ | India |
| 20 | →→→→→ →→→→ | →→→→ | India |
| 21 | →→→→→→→→→→→ | →→→→ | India |
| 22 | →→ →→→→ | →→→→→ →→→→→ | Israel |
| 23 | →→→→→→→→→→→ | →→→→→ →→→→→ | Israel |
| 24 | →→→→→→→ | →→→→→ →→→→→ | Israel |
| 25 | →→ | →→ | Japan |
| 26 | →→→ | →→ | Japan |
| 27 | →→ | →→ | Japan |
| 28 | →→ | →→ | Japan |
| 29 | →→ | →→ | Korea |
| 30 | →→ | →→ | Korea |
| 31 | →→ | →→ | Korea |
| 32 | Panev→◆ys | Lietuva | Lithuania |
| 33 | Tel◆iai | Lietuva | Lithuania |
| 34 | ◆iauliai | Lietuva | Lithuania |
| 35 | O→wi→cim | Polska | Poland |
| 36 | Warszawa | Polska | Poland |
| 37 | →◆d→ | Polska | Poland |

*Figure 2: Multilingual "ASCIIANY" data set*

However, here is a trick that enables you to do this. Provided you do not mind to see question marks for characters that you do not need, you can run the macro listed below. It uses the KPROPDATA function. This function removes or converts unprintable characters. The macro code enables you to transcode from and to any encoding; by default characters that are not representable in the target encoding are converted to question marks. KPROPDATA also enables you to select other options if characters are found in the data that are not supported by the encoding. For example, the characters can be preserved by converting them to Unicode escape (\Uxxxx) or NCR format. Or, you could instruct KPROPDATA to substitute a space. Expected input is the source and target data sets, and optionally the from and to encodings. If you omit from encoding specification, the session encoding is used. If both from and to encoding is omitted, then it copies data by just filtering binary data.
NOTE: If you select UESC or NCR as a substitution format, you need to adjust the column length to accommodate the extra bytes that might be needed to represent those characters.

```
%macro sas_iconv_dataset(in,out,from=UNDEFINED,to=UNDEFINED,sub='?',file_opt=);
data &out(encoding=asciiany);
set &in;
array cc (*) _character_;
do _N_=1 to dim(cc);
   cc(_N_)=kpropdata(cc(_N_),&sub,"&from","&to");
 end;
run;
%let lib=%scan(&out,1,%str(.));%let mem=%scan(&out,2,%str(.));
%if %length(&mem) = 0 %then %do;%let mem=&lib; %let lib=work;%end;
proc datasets lib=&lib nolist;
modify &mem / correctencoding=&to;
run;
quit;
%mend;
%sas_iconv_dataset(WORK.countries,WORK.countrieslt1,from=utf-8,to=wlatin1);
```



VIEWTABLE: Work.Countrieslt1

| | Region | Short Country name | English country name |
|---|---|---|---|
| 12 | Midi-Pyrénées | France | France |
| 13 | Provence-Alpes-Côte d'Azur | France | France |
| 14 | Rhône-Alpes | France | France |
| 15 | ????? | ????? | Greece |
| 16 | ?????? | ????? | Greece |
| 17 | ??????????? | ????? | Greece |
| 18 | ?????? ?? ??????? ????? | ???? | India |
| 19 | ??????? ?????? | ???? | India |
| 20 | ????? ???? | ???? | India |
| 21 | ??????????? | ???? | India |
| 22 | ?? ???? | ????? ????? | Israel |
| 23 | ??????????? | ????? ????? | Israel |
| 24 | ??????? | ????? ????? | Israel |
| 25 | ?? | ?? | Japan |
| 26 | ??? | ?? | Japan |
| 27 | ?? | ?? | Japan |
| 28 | ?? | ?? | Japan |
| 29 | ?? | ?? | Korea |
| 30 | ?? | ?? | Korea |
| 31 | ?? | ?? | Korea |
| 32 | Panev?žys | Lietuva | Lithuania |
| 33 | Telšiai | Lietuva | Lithuania |
| 34 | Šiauliai | Lietuva | Lithuania |
| 35 | O?wi?cim | Polska | Poland |
| 36 | Warszawa | Polska | Poland |
| 37 | ?ód? | Polska | Poland |

C:\Program Files\SASHome\SASFoundation\9.3

*Figure 3: Multilingual data set after KPROPDATA*

## (R)DBMS Tables

SAS/ACCESS® software enables SAS to share data with Oracle, DB2, and other relational database management systems (RDBMS). SAS can read and write data to a database management system (DBMS) in exactly the same way as reading from or writing to a SAS library.
Transcoding from the database (server) encoding to the client (application session) encoding is usually determined by the OS locale where the client is installed. However, some of the RDBMS clients do not depend on the OS and can be configured differently. This is also true for SAS. The trick, then, is to **make sure that the server knows the client's encoding. In any case, as long as the SAS session encoding is equivalent to the RDBMS client encoding value, there is no data loss.**
Each RDBMS client software application has its own way of being configured with the client application encoding. Client variables ensure that the transcoding from the database encoding to the client's encoding works correctly. See the following table for illustration.

| **RDBMS** | **RDBMS client variable(s)** | **Example** |
|---|---|---|
| DB2 | DB2CODEPAGE | DB2CODEPAGE 1208 |
| Greenplum | Windows: no options needed<br>UNIX: IANAAppCodePage | IANAAppCodePage=106 |
| Hadoop | No options needed;<br>The default value of the JRE Options file.encoding is UTF8 in a SAS Unicode session. If this value is changed, it should be modified as UTF8. Run proc javainfo all;run;to check this value in the log. | |
| Microsoft SQL Server | No options are needed; only nchar/nvarchar column is needed | |
| MySQL | names<br>or<br>character_set_client | set names 'UTF8'<br>or<br>set character_set_client=utf-8 |

| | | |
|---|---|---|
| | character_set_connection<br>character_set_results | set character_set_connection=utf-8<br>set character_set_results=utf-8 |
| Netezza | No options are needed; only nchar/nvarchar column is needed | |
| OLE DB | No options needed | |
| Oracle | NLS_LANG | NLS_LANG=American_America.AL32UTF8 |
| PostgreSQL | Windows: no options needed<br>UNIX: PGCLIENTENCODING | PGCLIENTENCODING=UTF8 |
| Sybase | locale | locale = default, us_english, utf8 |
| Teradata | charset_id<br>charset_type | charset_id=UTF8<br>charset_type=N |
| Vertica | No options needed. SAS/ACCESS handles the transcoding automatically | |

*Table 3: Client Variable Values of various RDBMSes*

Transcoding might fail if the source and target encodings are not (fully) compatible (that is, the destination encoding does not support all of the characters of the source encoding).

There are a couple of other attributes that you need to observe: which (character) data type the particular variables have and how variable lengths are defined. Most database management systems use the following data types to hold textual data: CHAR, VARCHAR, NCHAR, and NVARCHAR. NCHAR and NVARCHAR are used to store Unicode data; CHAR and VARCHAR are used to store non-Unicode character data. If your application was designed for one language or language group using the CHAR/VARCHAR data type is fine. But if working with multilingual data or if the server encoding is not compatible with the user data (encoding), Unicode data types, such as NCHAR/NVARCHAR are recommended.

Calculating column lengths in bytes is called BYTE semantics; measuring the lengths in characters is called CHARACTER semantics. Depending on the data type, most database management systems support either BYTE or CHARACTER semantics. Oracle is the only RDBMS that supports both BYTE and CHARACTER semantics. See the following table for illustration.

| **RDBMS** | | **Data type** | **Encoding of data** | **Length semantics** | **Notes** |
|---|---|---|---|---|---|
| DB2 | Character data | CHAR | ASCII[3] | Byte | Under the UNICODE database these data types can store UTF-8 data. |
| | | VARCHAR | | | |
| | | LONG VARCHAR | | | |
| | | GRAPHIC | KANJI | Byte | Under the UNICODE database these data types can store UCS-2 data. |
| | | LONG GRAPHIC | | | |
| | National Character data | not applicable | not applicable | not applicable | |
| Greenplum | Character Data | CHAR | Latin1, Shift-JIS, UTF-8. | Character | Character data types should be equivalent to PostgreSQL which the length semantics is character-based and can store Unicode data |
| | | VARCHAR | | | |
| | | TEXT | | | |
| | National Character data | CHAR | UTF-8 | Character | . UTF-8 CHAR is equivalent to NCHAR |
| | | VARCHAR | | | |
| | | TEXT | | | |
| Microsoft SQL Server | Character Data | CHAR | Latin1, Shift-JIS, UTF-8 . | Byte | |
| | | VARCHAR | | | |
| | | TEXT | | | |
| | | NCHAR | UTF-16 | Character | |
| | | NVARCHAR | | | |

[3] ASCII here means any ASCII-based encoding such as Latin1 or Shift-JIS.

| | | | | | |
|---|---|---|---|---|---|
| | National Character data | NTEXT | | | |
| MySQL | Character data | CHAR | LATIN, Shift-JIS, | Byte / Character | In MySQL 4.1 or later, length is based on Character semantics. |
| | | VARCHAR | | | |
| | | TEXT | | | |
| | | MEDIUMTEXT | | | |
| | | LONGTEXT | | | |
| | National Character data | NCHAR | UTF-8 | Character | |
| | | NVARCHAR | | | |
| Netezza | Character data | CHAR | Latin9[4] | Byte | |
| | | VARCHAR | | | |
| | National Character data | NCHAR | UTF-8[5] | Character | |
| | | NVARCHAR | | | |
| Oracle | Character data | CHAR | Latin1, Shift-JIS, UTF-8, . | Byte | With NLS_LENGTH_SEMANTICS=CHAR the length semantics becomes character. |
| | | VARCHAR2 | | | |
| | | CLOB | | | |
| | | LONG VARCHAR | | | |
| | National Character data | NCHAR | UTF-16 | Character | |
| | | NVARCHAR2 | | | |
| | | NCLOB | | | |
| PostgreSQL | Character data | CHAR | Latin1, Shift-JIS, | Character | |
| | | VARCHAR | | | |
| | | TEXT | | | |
| | National Character data | CHAR | UTF-8 | Character | UTF-8 CHAR is equivalent to NCHAR |
| | | VARCHAR | | | |
| | | TEXT | | | |
| Sybase | Character data | CHAR | Latin1, Shift-JIS, UTF-8 | Byte | |
| | | VARCHAR | | | |
| | | TEXT | | | |
| | National Character data | NCHAR | UTF-8 | Character | |
| | | NVARCHAR | | | |
| | | UNICHAR | UTF-16 | | available from ASE12.5 |
| | | UNIVARCHAR | | | |

*Table 4: Data types and length semantics*

The length semantics determine whether the length of a column is specified in bytes or in characters. For example, a definition of CHAR(3) in CHARACTER semantics means 3 character in length. In BYTE semantics, it means 3 bytes in length. This is not an issue for single-byte encodings where a character is always 1 byte. However, this difference does become an issue with multi-byte encodings where a character is not always equivalent to a byte.

Data truncation might occur when the destination data buffer is shorter than needed. In a (R)DBMS client/server environment, this might happen for two reasons:

- Data expansion due to transcoding. For example, when transcoding WLATIN1 data to UTF-8, some WLATIN1 characters are converted to 2 or 3-byte UTF-8 characters. Therefore, the caller needs to provide a buffer three times the size of the original buffer.
- Difference of length semantics between the (R)DBMS server and the client application. An example is the difference between NCHAR data and SAS character data. SAS/ACCESS products map the SAS character data type to DBMS data type when reading/writing data. SAS always uses BYTE LENGTH SEMANTICS while the length semantics of NCHAR is CHARACTER.

[4] The char/varchar data type only supports the ISO Latin-9 encoding.
[5] The ANSI SQL standard nchar/nvarchar data type supports Unicode using the UTF-8 encoding.

## Language Switching

The language switching feature enables you to produce multilingual reports on the fly by switching languages during the same SAS session. This feature is particularly interesting with a Unicode server, which can support many different languages. The language switching feature takes advantage of the existing localizations where SAS products have been localized, or translated into different languages. Localization of these products involves translating various components such as the message files system, ODS templates, SAS registry files, and other files. Language switching also affects the results of  locale sensitive formats, supplied by SAS, as well as some features SAS enables you to customize to produce locale-specific results.
Language switching was introduced in SAS 9.2 for the Unicode server. In SAS 9.4 it can also be used with legacy encodings.  For this purpose several new options were implemented:

- LOGLANGCHG Option. This option enables the language of message switching in SAS log output. If LOGLANGCHG is specified, the language of SAS log depends on LSWLANG or LOCALE= option. LSWLANG has the higher priority. If LSWLANG is set to a valid SAS language, SAS log output is controlled by the value of LSWLANG; otherwise, LOCALE= option determines the language of SAS log.
- ODSLANGCHG Option. This option enables the language of message switching in SAS ODS output. If ODSLANGCHG is ON, the language of ODS output depends on LSWLANG option or LOCALE= option. LSWLANG has the higher priority. If LSWLANG is set to a valid SAS language, ODS output is controlled by the value of LSWLANG; otherwise, LOCALE= option determines the language of ODS output.
- LSWLANG Option. It specifies the language of messages if LOGLANGCHG or ODSLANGCHG is ON. The default setting of LSWLANG is the string “LOCALE”, which means the LOCALE= option determines the language for switching. The setting of LSWLANG= option also needs to be compatible with SAS session encoding, otherwise its value is ignored.
- LOGLANGENG Option. It is a toggle option that overrides LOGLANGCHG and LSWLANG and sets them to LOGLANGCHG=ON, LSWLANG=”EN”. As result, the LOG output is in  English, and NL Format output has no change. This option does change the setting of ODSLANGCHG. If ODSLANGCHG=OFF, the system message language for ODS output is determined by SAS configuration: If ODSLANGCHG=ON, all messages are in English because of the LSWLANG setting.
- LOCALELANGCHG Option. LOGCALELANGCHG= option was introduced in SAS 9.2, and acts an alias of ODSLANGCHG in SAS 9.4.

Please note that language switching in a legacy environment only makes sense with languages that use the same encoding. For example, Western European languages such as French, German, Italian, Spanish, and Swedish all share the Latin1 character repertoire; Central and Eastern European languages such as Czech, Hungarian and Polish share the Latin2 character repertoire. Hence, in a legacy environment, you can switch between French and German, or between Hungarian and Polish but not between French and Polish. Switching between Asian (CJK) languages in a legacy environment does not make sense at all. Of course, in a Unicode server (UTF-8) session, you can switch between any languages.

NOTE:  Language switching is **not supported on z/OS**.

In order to use the language switching feature, in the SAS program examples below, the SAS System has been started with the following options at start-up: *sas –lswlang locale –odslangchg –nologlangchg*

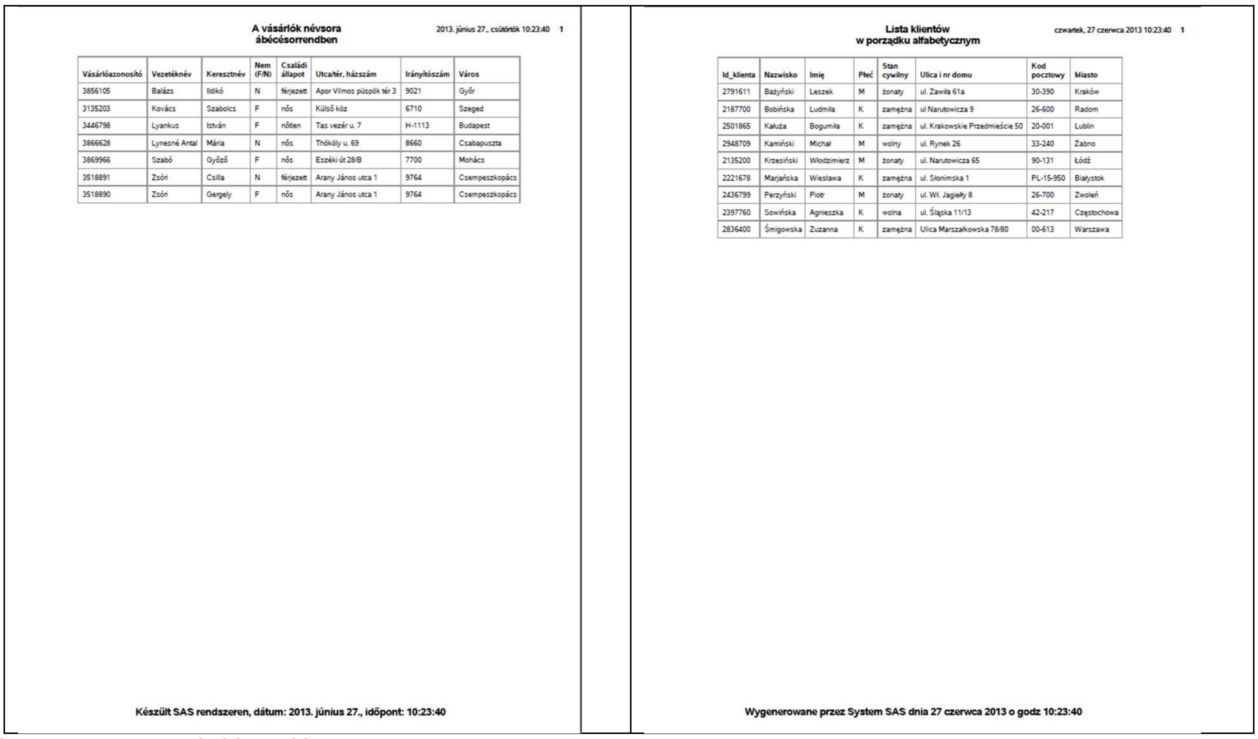
This means that the LOCALE= option determines the language for switching and the language of ODS output.  The language of SAS log is the one used at system start.
Let us suppose that we create lists of customers from a Latin2 marketing campaign database. Depending on the users’ preferred language that we create separate lists of Hungarian and Polish-speaking customers. For output, we sort these lists according to local language conventions using SORTSEQ=LINGUISTIC.

```
/* ------------------------------------------------------------------
   Create subsets of the data
   ------------------------------------------------------------------ */
proc sql;
create table polish as
select customer_ID, name, first, gender, marital_status, street, zip, city from
t.new_contacts where country_e = 'Poland';
create table hungarian as
select customer_ID, name, first, gender, marital_status, street, zip, city from
t.new_contacts where country_e = 'Hungary';
quit;
```

```
/* ------------------------------------------------------------------
   label variables in different languages
   ------------------------------------------------------------------ */
data polish;
   set polish;
   label customer_ID='Id_klienta';
   label gender='Płeć';
   label marital_status='Stan cywilny';
   label name='Nazwisko';
   label first='Imię';
   label street='Ulica i nr domu';
   label city='Miasto';
   label zip='Kod pocztowy';
run;
data hungarian;
   set hungarian;
   label customer_ID='Vásárlóazonosító';
   label gender='Nem (F/N)';
   label marital_status='Családi állapot';
   label name='Vezetéknév';
   label first='Keresztnév';
   label street='Utca/tér, házszám';
   label city='Város';
   label zip='Irányítószám';
run;

/* ------------------------------------------------------------------
   Run the SORT Procedure and create PDF output
   ------------------------------------------------------------------ */
ods pdf file='c:\temp\polish.pdf';
options locale=pl_PL dflang=locale;
TITLE;
TITLE1 "Lista klientów ";
TITLE2 "w porządku alfabetycznym";
FOOTNOTE;
FOOTNOTE1 "Wygenerowane przez System SAS dnia %TRIM(%QSYSFUNC(DATE(), NLDATE20.)) o
godz %SYSFUNC(TIME(), NLTIME10.)";
proc sort data=polish sortseq=linguistic;
  by name;
run;
proc print data=polish label noobs; run;
RUN; QUIT;
TITLE; FOOTNOTE;
/* ------------------------------------------------------------------
   DTRESET specifies that SAS update the date and time in the titles
   of the SAS log and the listing file.
   ------------------------------------------------------------------ */
OPTIONS DTRESET;
/* ------------------------------------------------------------------
   Run the SORT Procedure and create PDF output
   ------------------------------------------------------------------ */
ods pdf file='c:\temp\hungarian.pdf';
options locale=hu_HU dflang=locale;
TITLE;
TITLE1 "A vásárlók névsora ";
TITLE2 "ábécésorrendben";
FOOTNOTE;
FOOTNOTE1 "Készült SAS rendszeren, dátum: %TRIM(%QSYSFUNC(DATE(), NLDATE20.)),
időpont: %SYSFUNC(TIME(), NLTIME10.)";
proc sort data=hungarian sortseq=linguistic;
  by name;
run;
```

```
proc print data=hungarian label noobs; run;
RUN; QUIT;
TITLE; FOOTNOTE;
ods pdf close;
```

Output looks like the following:



A vásárlók névsora
ábécésorrendben

2013. június 27., csütörtök 10:23:40 1

| Vásárlóazonosító | Vezetéknév | Keresztnév | Nem (F/N) | Családi állapot | Utca/tér, házszám | Irányítószám | Város |
|---|---|---|---|---|---|---|---|
| 3856105 | Balázs | Ildikó | N | férjezett | Apor Vilmos püspök tér 3 | 9021 | Győr |
| 3135203 | Kovács | Szabolcs | F | nős | Külső köz | 6710 | Szeged |
| 3446798 | Lyankus | István | F | nőtlen | Tas vezér u. 7 | H-1113 | Budapest |
| 3866628 | Lynesné Antal | Mária | N | nős | Thököly u. 69 | 8660 | Csabapuszta |
| 3869966 | Szabó | Győző | F | nős | Eszéki út 28/B | 7700 | Mohács |
| 3518891 | Zsóri | Csilla | N | férjezett | Arany János utca 1 | 9764 | Csempeszkopács |
| 3518890 | Zsóri | Gergely | F | nős | Arany János utca 1 | 9764 | Csempeszkopács |

Készült SAS rendszeren, dátum: 2013. június 27., időpont: 10:23:40

Lista klientów
w porządku alfabetycznym

czwartek, 27 czerwca 2013 10:23:40 1

| Id_klienta | Nazwisko | Imię | Płeć | Stan cywilny | Ulica i nr domu | Kod pocztowy | Miasto |
|---|---|---|---|---|---|---|---|
| 2791611 | Bażyński | Leszek | M | żonaty | ul. Zawiła 61a | 30-390 | Kraków |
| 2187700 | Bobińska | Ludmiła | K | zamężna | ul Narutowicza 9 | 26-600 | Radom |
| 2501865 | Kałuża | Bogumiła | K | zamężna | ul. Krakowskie Przedmieście 50 | 20-001 | Lublin |
| 2948709 | Kamiński | Michał | M | wolny | ul. Rynek 26 | 33-240 | Żabno |
| 2135200 | Krzesiński | Włodzimierz | M | żonaty | ul. Narutowicza 65 | 90-131 | Łódź |
| 2221678 | Marjańska | Wiesława | K | zamężna | ul. Słonimska 1 | PL-15-950 | Białystok |
| 2436799 | Perzyński | Piotr | M | żonaty | ul. Wł. Jagiełły 8 | 26-700 | Zwoleń |
| 2397760 | Sowińska | Agnieszka | K | wolna | ul. Śląska 11/13 | 42-217 | Częstochowa |
| 2836400 | Śmigowska | Zuzanna | K | zamężna | Ulica Marszałkowska 78/80 | 00-613 | Warszawa |

Wygenerowane przez System SAS dnia 27 czerwca 2013 o godz 10:23:40

*Table 5: Language Switching with PDF output*

If you want to create output using the localized SAS components such as ODS templates from the FREQ procedure you need to make sure the appropriate catalogs are available. This means you have to update the SAS config file and insert the path to the localized sashelp[6]; e.g.:

```
-SASHELP (
      "!SASCFG\SASCFG"
      "!SASROOT\nls\fr\sashelp"
      "!SASROOT\nls\it\sashelp"
      "!SASROOT\nls\de\sashelp"
      "!SASROOT\core\sashelp"
```

This enables you to use the localized files for French, Italian and German, for example.
To evaluate a Latin1 marketing campaign database statistically with PROC FREQ, for example, you could run the code below that is creating separate output for French, German and Italian-speaking customers in Switzerland:

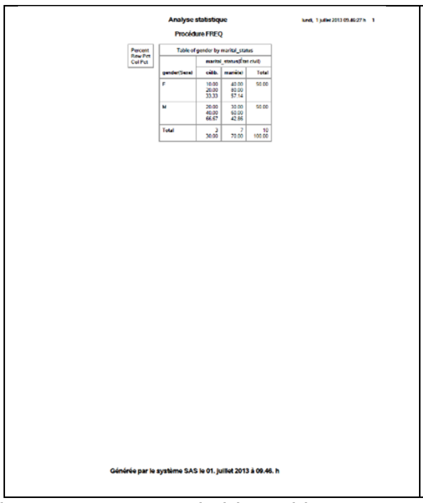
```
/* ------------------------------------------------------------------
   Create subsets of the data
   ------------------------------------------------------------------ */

proc sql;
create table french as
select customer_ID, name, first, gender, marital_status, street, zip, city from
t.swiss where lang = 'fra';
create table german as
select customer_ID, name, first, gender, marital_status, street, zip, city from
t.swiss where lang = 'deu';
create table italian as
```

[6] No update for the SAS configuration file is needed when using SAS Unicode server.

```
select customer_ID, name, first, gender, street, marital_status, zip, city from
t.swiss where lang = 'ita';
quit;

/* ------------------------------------------------------------------
   label variables in different languages
   ------------------------------------------------------------------ */
data french;
   set french;
   label customer_ID='Numéro de client';
   label name='Nom';
   label first='Prénom';
   label gender='Sexe';
   label marital_status='État civil';
   label street='Adresse';
   label city='Lieu';
   label zip='CP';
run;

data german;
   set german;
   label customer_ID='Kundennummer';
   label name='Name';
   label first='Vorname';
   label gender='Geschlecht';
   label marital_status='Familienstand';
   label street='Strasse';
   label city='Ort';
   label zip='PLZ';
run;

data italian;
   set italian;
   label customer_ID='Numero del cliente';
   label name='Nome';
   label first='Cognome';
   label gender='Sesso';
   label marital_status='Stato civile';
   label street='Strada';
   label city='Città';
   label zip='CAP';
run;
/* ------------------------------------------------------------------
   Create PDF output
   ------------------------------------------------------------------ */
ods pdf file='c:\temp\french.pdf';
options locale=fr_CH dflang=locale;
TITLE;
TITLE1 "Analyse statistique";

FOOTNOTE;
FOOTNOTE1 "Générée par le système SAS le %TRIM(%QSYSFUNC(DATE(), NLDATE20.))
à %SYSFUNC(TIME(), NLTIME10.)";

proc freq data=french;
   Tables gender*marital_status / NOFREQ NOCUM;

RUN; QUIT;

TITLE; FOOTNOTE;
/* ------------------------------------------------------------------
   DTRESET specifies that SAS update the date and time in the titles
```

```
      of the SAS log and the listing file.
      ------------------------------------------------------------------- */
   OPTIONS DTRESET;

   ods pdf file='c:\temp\german.pdf';
   options locale=de_CH dflang=locale;
   TITLE;
   TITLE1 "Statistische Auswertung";

   FOOTNOTE;
   FOOTNOTE1 "Generiert durch das SAS System am %TRIM(%QSYSFUNC(DATE(), NLDATE20.))
   um %SYSFUNC(TIME(), NLTIME10.)";


   proc freq data=german;
      Tables gender*marital_status / NOFREQ NOCUM;

   RUN; QUIT;
   TITLE; FOOTNOTE;
   /* -------------------------------------------------------------------
      DTRESET specifies that SAS update the date and time in the titles
      of the SAS log and the listing file.
      ------------------------------------------------------------------- */
   OPTIONS DTRESET;

   ods pdf file='c:\temp\italian.pdf';
   options locale=it_CH dflang=locale;
   TITLE;
   TITLE1 "Analisi";
   FOOTNOTE;
   FOOTNOTE1 "Generato dal sistema SAS il %TRIM(%QSYSFUNC(DATE(), NLDATE20.))
   alle %SYSFUNC(TIME(), NLTIME10.)";


   proc freq data=italian;
      Tables gender*marital_status / NOFREQ NOCUM;

   RUN; QUIT;
   TITLE; FOOTNOTE;
   ods pdf close;
```

This is what the output looks like:



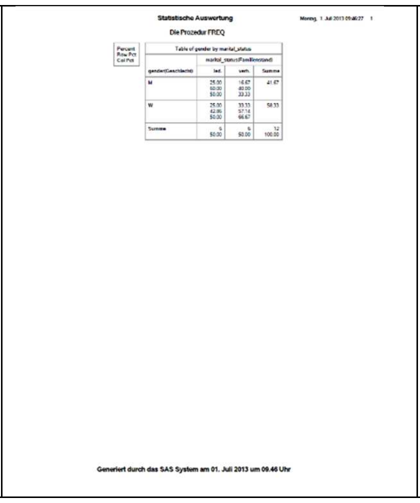
Analyse statistique — lundi, 1 juillet 2013 09.46:27 h 1

Procédure FREQ

Percent
Row Pct
Col Pct

| Table of gender by marital_status | | | |
|---|---|---|---|
| | marital_status(État civil) | | |
| gender(Sexe) | célib. | marié(e) | Total |
| F | 10.00<br>20.00<br>33.33 | 40.00<br>80.00<br>57.14 | 50.00 |
| M | 20.00<br>40.00<br>66.67 | 30.00<br>60.00<br>42.86 | 50.00 |
| Total | 3<br>30.00 | 7<br>70.00 | 10<br>100.00 |

Généré par le système SAS le 01. juillet 2013 à 09.46. h



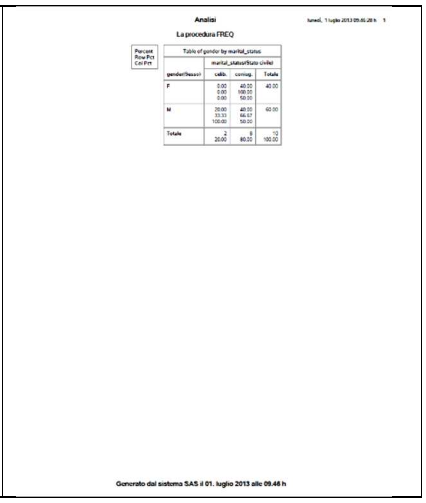
Statistische Auswertung — Montag, 1. Juli 2013 09:46:27 1

Die Prozedur FREQ

Percent
Row Pct
Col Pct

| Table of gender by marital_status | | | |
|---|---|---|---|
| | marital_status(Familienstand) | | |
| gender(Geschlecht) | led. | verh. | Summe |
| M | 25.00<br>60.00<br>50.00 | 16.67<br>40.00<br>33.33 | 41.67 |
| W | 25.00<br>42.86<br>50.00 | 33.33<br>57.14<br>66.67 | 58.33 |
| Summe | 6<br>50.00 | 6<br>50.00 | 12<br>100.00 |

Generiert durch das SAS System am 01. Juli 2013 um 09.46 Uhr



Analisi — lunedì, 1 luglio 2013 09.46.28 h 1

La procedura FREQ

Percent
Row Pct
Col Pct

| Table of gender by marital_status | | | |
|---|---|---|---|
| | marital_status(Stato civile) | | |
| gender(Sesso) | celib. | coniug. | Totale |
| F | 0.00<br>0.00<br>0.00 | 40.00<br>100.00<br>50.00 | 40.00 |
| M | 20.00<br>33.33<br>100.00 | 40.00<br>66.67<br>50.00 | 60.00 |
| Totale | 2<br>20.00 | 8<br>80.00 | 10<br>100.00 |

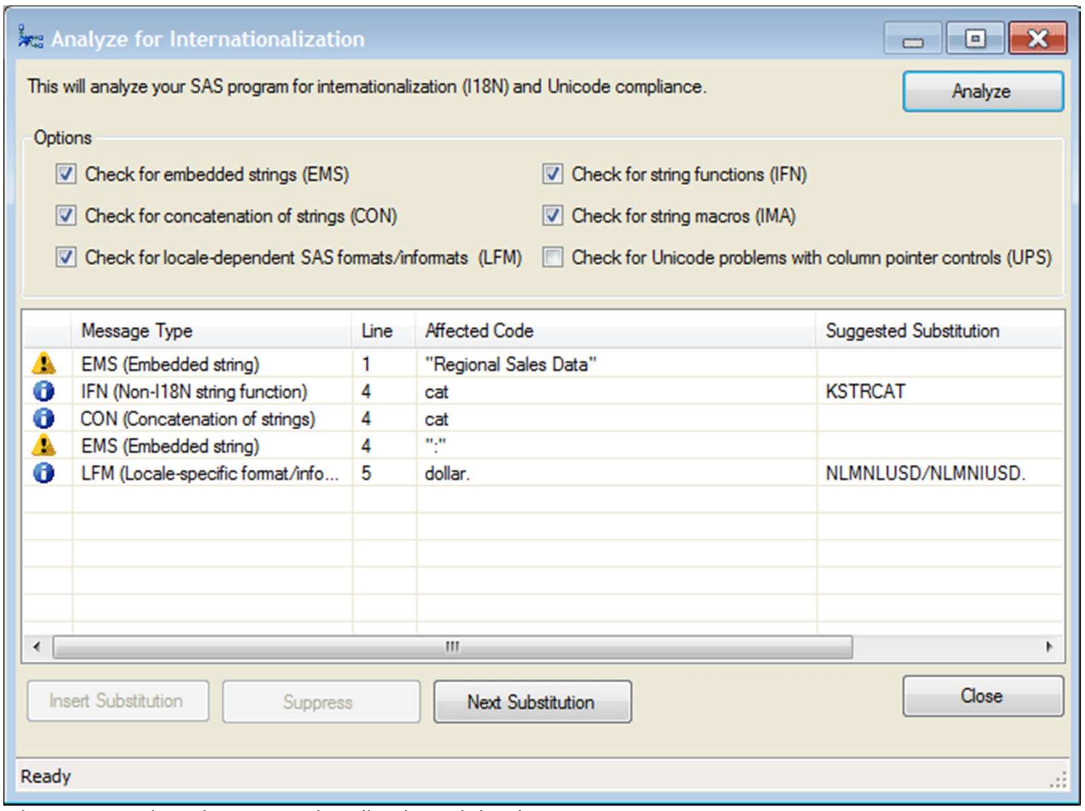
Generato dal sistema SAS il 01. luglio 2013 alle 09.46 h

*Table 6: Language Switching with PROC FREQ*

## Making Your SAS Code Locale-Sensitive

SAS ® Enterprise Guide ® 6.1 ships a new feature that helps customers determine if there are any possible internationalization issues within their own SAS code. Internationalization, often abbreviated as I18n, is the process by which a program is optimized so that it can be adapted to any locale without needing to be rewritten. When you analyze a program for internationalization, SAS ® Enterprise Guide ® lists the lines of code that might be affected and suggests a substitution when possible. It makes suggestions for substitutions of the SAS supplied (intrinsic) formats and string functions and will even replace those in the SAS code. It will also point out embedded text.
To analyze a program for internationalization:

1. Open the program in the SAS Enterprise Guide workspace.
2. Select **Program** ▸ **Analyze** ▸ **Analyze Program for I18n Issues**. The Analyze for Internationalization dialog box opens.
3. Select the check boxes for the types of issues that you want to include in the analysis. By default, all of the check boxes are selected except **Check for Unicode problems with column pointer controls (UPS)**.
4. Click **Analyze**. The results show each affected line of code in the program. You can click a line in the Analyze for Internationalization dialog box to highlight the affected line in the Program window.
5. When you view the results of the analysis, you can select the following options:
    - **Insert Substitution** - replaces the affected code in your program with the suggested substitution. This option is available only if SAS Enterprise Guide has a suggested substitution.
    - **Suppress** - inserts a comment on the affected line of code so that it is excluded from analysis.
    - **Next Substitution** - advances to the next substitution.



This will analyze your SAS program for internationalization (I18N) and Unicode compliance.

- [✓] Check for embedded strings (EMS)
- [✓] Check for string functions (IFN)
- [✓] Check for concatenation of strings (CON)
- [✓] Check for string macros (IMA)
- [✓] Check for locale-dependent SAS formats/informats (LFM)
- [ ] Check for Unicode problems with column pointer controls (UPS)

| | Message Type | Line | Affected Code | Suggested Substitution |
|---|---|---|---|---|
| ⚠ | EMS (Embedded string) | 1 | "Regional Sales Data" | |
| ℹ | IFN (Non-I18N string function) | 4 | cat | KSTRCAT |
| ℹ | CON (Concatenation of strings) | 4 | cat | |
| ⚠ | EMS (Embedded string) | 4 | ":" | |
| ℹ | LFM (Locale-specific format/info... | 5 | dollar. | NLMNLUSD/NLMNIUSD. |

*Figure 4: Analyze for Internationalization Dialog in EG*

When you analyze a program for internationalization, SAS Enterprise Guide checks for the following items:

- Embedded strings (EMS) - text strings that are embedded in a program allow the application to display only one language to the user. Any strings that are visible in the user interface should be externalized, or stored in a separate properties file, so they can be translated. For information about replacing strings in your SAS program, see the help for the SASMSG function in the *SAS 9.4 National Language Support (NLS): Reference Guide*.
- Concatenated text strings (CON) - concatenation is the operation in which two or more text strings are combined into one string. However, since grammatical structures of language can be very different, concatenation of strings can cause grammatical conflicts. It is best to avoid concatenation of strings that you intend to translate.
- Locale-dependent SAS formats/informats (LFM) - formats and informats can present locale dependencies for types of data including dates, time, currency, and numerical values. You can use SAS NL formats to avoid such locale-specific dependencies.
- String functions (IFN) - SAS provides many string functions and call routines that can be used to manipulate characters and strings. Some string functions in SAS assume that the size of a single character is always one byte. In some environments, however, a single character can be up to four bytes. SAS provides additional K string functions that do not make assumptions about the size of a single character. Regular functions are byte-oriented, but K functions are character-oriented (as explained above).
- String macros (IMA) - SAS provides many macros that can be used to manipulate characters and strings. Some string macros in SAS assume that the size of a single character is always one byte. In some environments, however, a single character can be up to four bytes. SAS provides additional NLS macro functions that do not make assumptions about the size of a single character.
- Column pointer controls in the PUT and INPUT statements in a UTF-8 SAS session (UPS)- the SAS column pointer controls assume that each character can be displayed in one column. However, a character in a multi-byte encoding, such as UTF-8, is not equivalent to one byte as characters are in single-byte encodings. In UTF-8 encoding, all non-ASCII characters have a varying length between two and four bytes, which can cause the SAS column pointer controls to return unexpected results.

To exclude code from analysis

When you analyze your code, you might find that some of the reported issues occur in code that does not need to be changed. You can prevent SAS Enterprise Guide from including those issues by adding a comment to a single line or around a block of code. When you add the comment to your code, you can specify a particular issue that you want to exclude from the analysis, or you can exclude all issues. For example, you can exclude embedded strings (EMS) from the analysis of one line of code as follows:

```
data a;
 x="text string"; /* ignore for i18n analyzer - ID:EMS */
run;
```

You can exclude an entire block of code from all analysis as follows:

```
/*  ignore for i18n analyzer - ID:ALL:begin */
data a;
  x='string 1';
  y='string 2';
  z=today();
  put z date.;
run;
/*  ignore for i18n analyzer - ID:ALL:end */
```

More information about how to successfully internationalize your SAS programs and making applications "locale-aware" is available in the SAS Global Forum paper "Give Some International Flavor to Your SAS® Applications".

## Multilingual Support in SAS® Visual Analytics

SAS Visual Analytics not only leverages the in-memory analytic engine, the SAS LASR Analytic Server, letting you visually explore big data at fast speeds, but also provides support for multiple languages, enabling people from different regions in your organization to run analytics on multiple language data, and produce reports in their language and according to their conventions.

Multilingual language data can easily be loaded into SAS LASR Analytic Server without any additional transcoding. After that, you can perform data exploration and create reports on these data with SAS VA applications in your language.
The user interfaces, number and date formats in the explorations and reports can switch automatically according to your language and conventions. At the same time, when you view reports on mobile devices or through web browsers, the number and date formats will switch automatically as well.
Besides the above features, SAS Visual Analytics also provides a way to easily create the same style reports for different languages.

Data sources in various languages can be loaded into SAS LASR Analytic Server, as long as the SAS session encoding is compatible with the data source encoding. For example, on a Linux operating system, use a Latin9 SAS session encoding to load French data and a EUC-CN SAS session encoding to load Chinese data. Of course, all data can be loaded into SAS LASR Analytic Server with an utf-8 SAS session.

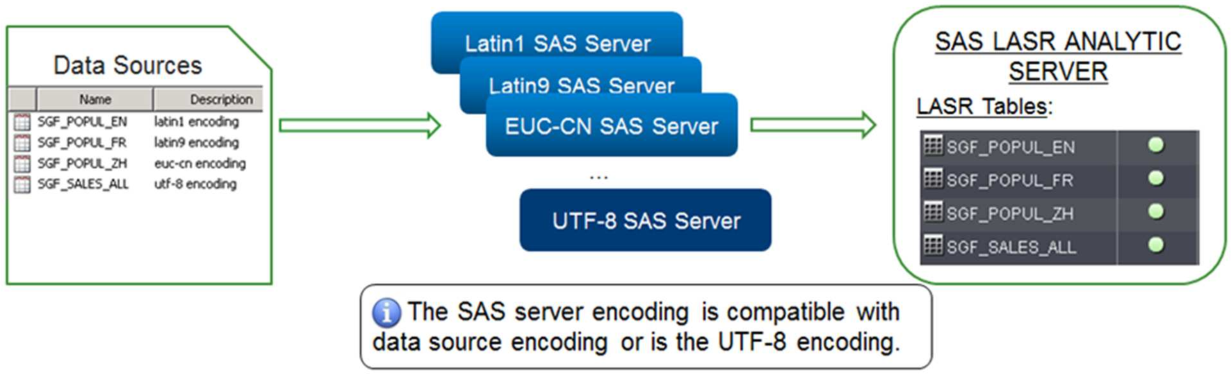

Data Sources
Name
Description
SGF_POPUL_EN latin1 encoding
SGF_POPUL_FR latin9 encoding
SGF_POPUL_ZH euc-cn encoding
SGF_SALES_ALL utf-8 encoding
Latin1 SAS Server
Latin9 SAS Server
EUC-CN SAS Server
...
UTF-8 SAS Server
SAS LASR ANALYTIC SERVER
LASR Tables:
SGF_POPUL_EN
SGF_POPUL_FR
SGF_POPUL_ZH
SGF_SALES_ALL
The SAS server encoding is compatible with data source encoding or is the UTF-8 encoding.

*Figure 5: Loading Data into SAS LASR Analytical Server*

After the data has been loaded into SAS LASR Analytic Server, you are able to explore and use it to create reports, no matter what language your SAS Visual Analytics application is using.

The language of the SAS Visual Analytics applications can be set to be the same as your browser language or set to a specific language. For example, you can explore English and Chinese data with an English SAS Visual Analytics application or explore the same Chinese data with an English or French SAS Visual Analytics application. All characters are displayed correctly.

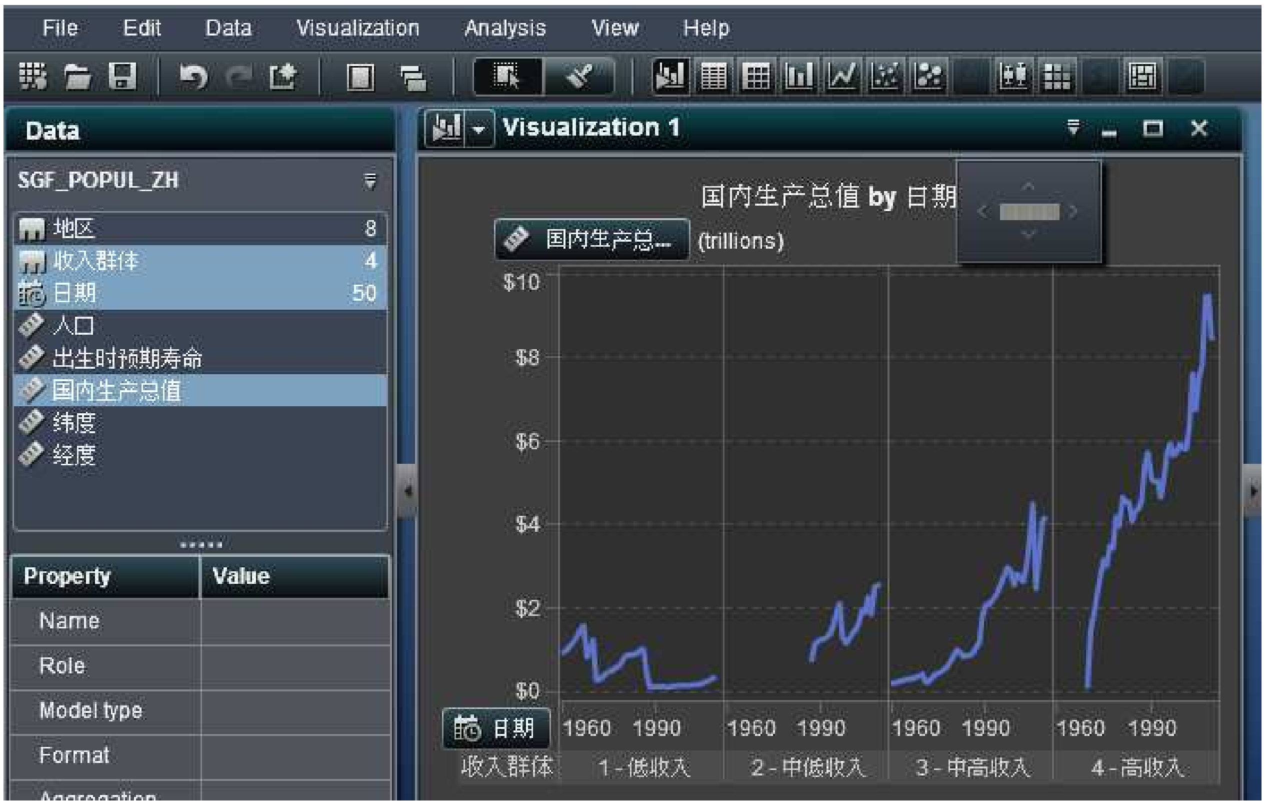
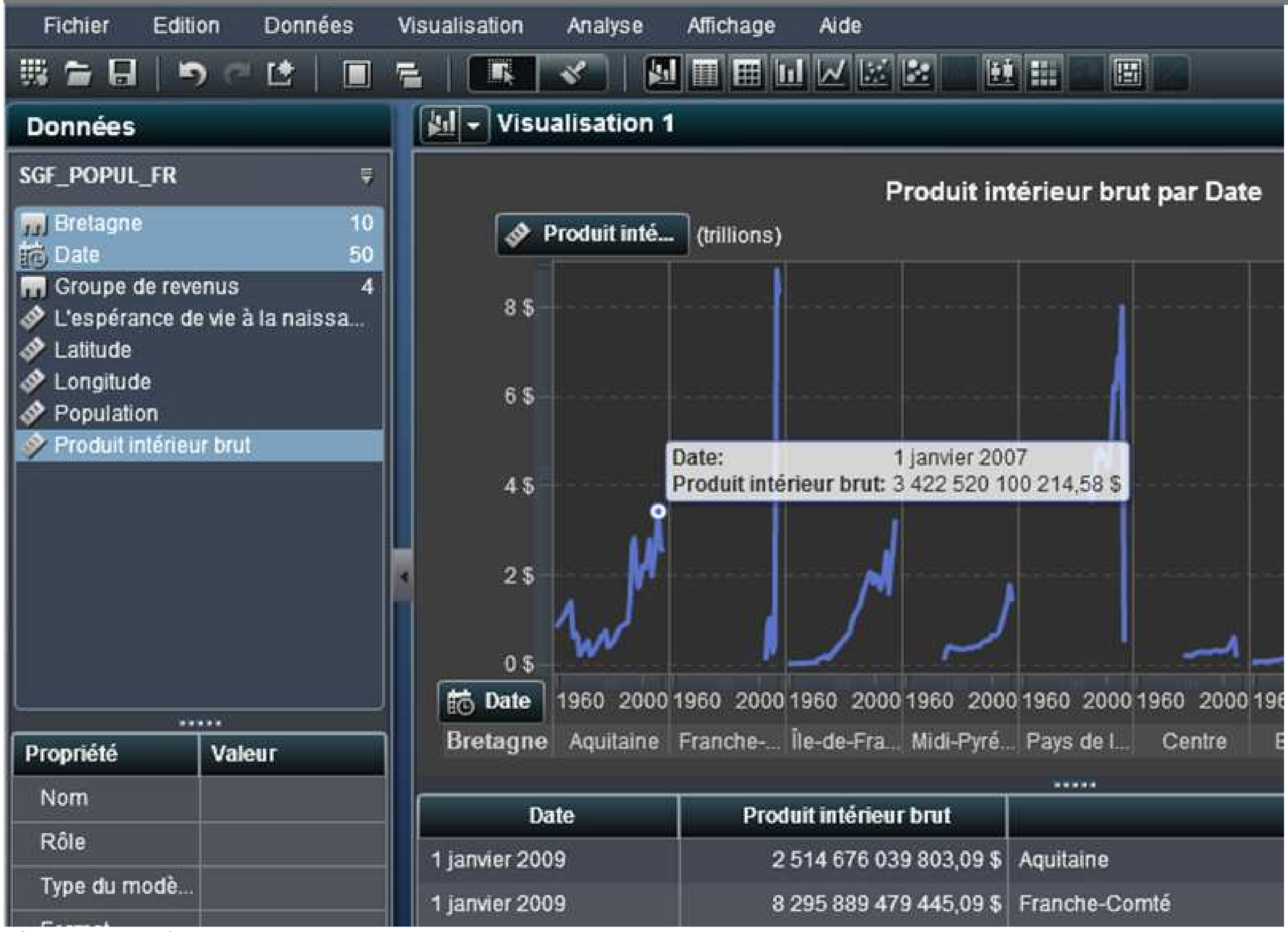


*Figure 6: English VA with Chinese data*

Moreover, the user interfaces, number and date formats of explorations and reports can switch automatically according to your VA application language. For example, when you are working with an English VA application: the user interfaces are in English, the dates and numbers are displayed according to English language and conventions.
When you are working with a French VA application: the user interfaces are in French; the display of dates and numbers change accordingly. You can see, in the figure below, the formats of dates and numbers are obviously different.



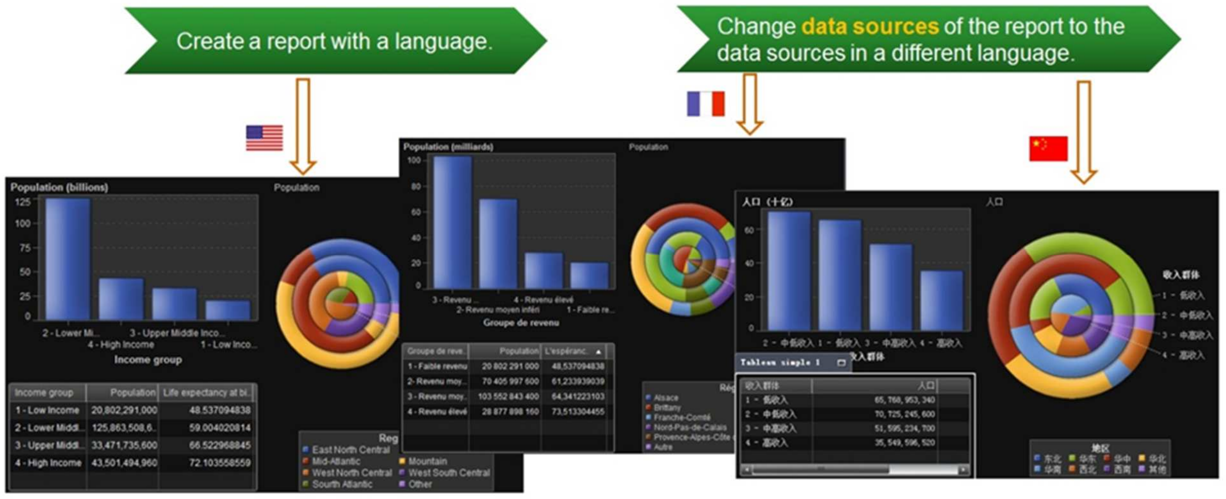
| | |
|---|---|
| Bretagne | 10 |
| Date | 50 |
| Groupe de revenus | 4 |
| L'espérance de vie à la naissa... | |
| Latitude | |
| Longitude | |
| Population | |
| Produit intérieur brut | |

| Propriété | Valeur |
|---|---|
| Nom | |
| Rôle | |
| Type du modè... | |

| Date | Produit intérieur brut | |
|---|---|---|
| 1 janvier 2009 | 2 514 676 039 803,09 $ | Aquitaine |
| 1 janvier 2009 | 8 295 889 479 445,09 $ | Franche-Comté |

*Figure 7: French VA*

SAS Visual Analytics is able to manipulate multiple language data simultaneously. Sometimes, it is necessary to create the same style reports for data sources in different languages. SAS Visual Analytics makes this easy to implement. Below is a report based on English data. It is very easy to propagate this report to data in another language.



| Income group | Population | Life expectancy at bi... |
|---|---|---|
| 1 - Low Income | 20,802,291,000 | 48.537094838 |
| 2 - Lower Middl... | 125,863,508,6... | 59.004020814 |
| 3 - Upper Middl... | 33,471,735,600 | 66.522968845 |
| 4 - High Income | 43,501,494,960 | 72.103558559 |

| Groupe de reve... | Population | L'espéranc... |
|---|---|---|
| 1 - Faible revenu | 20 802 291 000 | 48,537094838 |
| 2- Revenu moy... | 70 405 997 600 | 61,233939039 |
| 3 - Revenu moy... | 103 552 843 400 | 64,341223103 |
| 4 - Revenu élevé | 28 877 898 160 | 73,513304455 |

| 收入群体 | 人口 |
|---|---|
| 1 - 低收入 | 65, 768, 953, 340 |
| 2 - 中低收入 | 70, 725, 245, 600 |
| 3 - 中高收入 | 51, 595, 234, 700 |
| 4 - 高收入 | 35, 549, 596, 520 |

*Figure 8: Propagating a report in different languages*

You just need change the data sources of this report to data in another language. The report will display the information of the new data. When you view this report on mobile devices or web browser with this language, the report will be shown in the expected format. In the example above a French report and a Chinese report were generated this way. The prerequisite of this approach is that data sources in different languages have the same data items. This approach can save report design time and reduce repetitive workload.

## Conclusion

SAS 9.4 provides a number of new or enhanced features for handling multiple languages.  For broadest multilingual support, use of Unicode server is required.  However, you can use multiple languages and switch between them with legacy encodings as well – as long as those languages use a compatible encoding.

## Acknowledgments

Many thanks to all of those reviewers for their valuable feedback and for going over the technical details of the paper: Junichi Arai, Mickaël Bouedo, Troy Harris, Mark Kernodle, Joji Kobayashi, John Kohl, Helen (Honglian) Pan, Masahiro Takagi, and Tony (Tao) Zhang. Special thanks go to the editor Rachel Wynne, to Janusz Wojdecki and Csaba Prencsovszky for providing the Polish and Hungarian translations, and to Elizabeth Bales whose constructive comments improved each draft.

## References

Bouedo,  Mickaël and Beatrous, Stephen. 2013. *"Internationalization 101: Give Some International Flavor to Your SAS® Applications ."* *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings13/025-2013.pdf.

Kiefer, Manfred. 2012. *SAS® Encoding: Understanding the Details.* Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2013. *SAS/ACCESS® 9.4 for Relational Databases: Reference, Second Edition*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2013. *SAS® 9.4 Language Reference: Concepts, Second Edition*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2013. *SAS® 9.4 National Language Support (NLS): Reference Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2009. SAS Institute technical paper. "*Processing Multilingual Data with the SAS 9.2 Unicode Server*." Available at http://support.sas.com/resources/papers/92unicodesrvr.pdf**.**

## Appendix

```
# @(#)linux.fontconfig.RedHat.properties      1.11 10/03/23
#
# Copyright (c) 2003, Oracle and/or its affiliates. All rights reserved.
#

# Version

version=1

# Component Font Mappings

allfonts.chinese-cn-iso10646=-misc-zysong18030-medium-r-normal--*-%d-*-*-c-*-iso10646-1
allfonts.chinese-tw-iso10646=-misc-ar pl shanheisun uni-medium-r-normal--*-%d-*-*-c-*-iso10646-1
allfonts.lucida=-b&h-lucidasans-medium-r-normal-sans-*-%d-*-*-p-*-iso8859-1

serif.plain.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.plain.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.plain.latin-1=-b&h-lucidabright-medium-r-normal--*-%d-*-*-p-*-iso8859-1

serif.bold.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.bold.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.bold.latin-1=-b&h-lucidabright-demibold-r-normal--*-%d-*-*-p-*-iso8859-1

serif.italic.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.italic.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.italic.latin-1=-b&h-lucidabright-medium-i-normal--*-%d-*-*-p-*-iso8859-1

serif.bolditalic.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.bolditalic.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
serif.bolditalic.latin-1=-b&h-lucidabright-demibold-i-normal--*-%d-*-*-p-*-iso8859-1
```

```
sansserif.plain.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.plain.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.plain.latin-1=-b&h-lucidasans-medium-r-normal-sans-*-%d-*-*-p-*-iso8859-1

sansserif.bold.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.bold.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.bold.latin-1=-b&h-lucidasans-bold-r-normal-sans-*-%d-*-*-p-*-iso8859-1

sansserif.italic.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.italic.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.italic.latin-1=-b&h-lucidasans-medium-i-normal-sans-*-%d-*-*-p-*-iso8859-1

sansserif.bolditalic.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.bolditalic.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
sansserif.bolditalic.latin-1=-b&h-lucidasans-bold-i-normal-sans-*-%d-*-*-p-*-iso8859-1

monospaced.plain.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.plain.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.plain.latin-1=-b&h-lucidatypewriter-medium-r-normal-sans-*-%d-*-*-m-*-iso8859-1

monospaced.bold.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.bold.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.bold.latin-1=-b&h-lucidatypewriter-bold-r-normal-sans-*-%d-*-*-m-*-iso8859-1

monospaced.italic.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.italic.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.italic.latin-1=-b&h-lucidatypewriter-medium-i-normal-sans-*-%d-*-*-m-*-iso8859-1

monospaced.bolditalic.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.bolditalic.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
monospaced.bolditalic.latin-1=-b&h-lucidatypewriter-bold-i-normal-sans-*-%d-*-*-m-*-iso8859-1

dialog.plain.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.plain.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.plain.latin-1=-b&h-lucidasans-medium-r-normal-sans-*-%d-*-*-p-*-iso8859-1

dialog.bold.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.bold.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.bold.latin-1=-b&h-lucidasans-bold-r-normal-sans-*-%d-*-*-p-*-iso8859-1

dialog.italic.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.italic.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.italic.latin-1=-b&h-lucidasans-medium-i-normal-sans-*-%d-*-*-p-*-iso8859-1

dialog.bolditalic.japanese-iso10646=-misc-sazanami gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.bolditalic.korean-iso10646=-misc-baekmuk gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialog.bolditalic.latin-1=-b&h-lucidasans-bold-i-normal-sans-*-%d-*-*-p-*-iso8859-1

dialoginput.plain.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.plain.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.plain.latin-1=-b&h-lucidatypewriter-medium-r-normal-sans-*-%d-*-*-m-*-iso8859-1

dialoginput.bold.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.bold.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.bold.latin-1=-b&h-lucidatypewriter-bold-r-normal-sans-*-%d-*-*-m-*-iso8859-1

dialoginput.italic.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.italic.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.italic.latin-1=-b&h-lucidatypewriter-medium-i-normal-sans-*-%d-*-*-m-*-iso8859-1

dialoginput.bolditalic.japanese-iso10646=-misc-sazanami mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.bolditalic.korean-iso10646=-misc-baekmuk batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1
dialoginput.bolditalic.latin-1=-b&h-lucidatypewriter-bold-i-normal-sans-*-%d-*-*-m-*-iso8859-1

# Search Sequences

sequence.allfonts=latin-1
sequence.allfonts.UTF-8.ko.KR=latin-1,korean-iso10646
sequence.allfonts.UTF-8.ja.JP=latin-1,japanese-iso10646
sequence.allfonts.UTF-8.zh.CN=latin-1,chinese-cn-iso10646
sequence.allfonts.UTF-8.zh.TW=latin-1,chinese-tw-iso10646
sequence.allfonts.UTF-8.zh.HK=latin-1,chinese-tw-iso10646
sequence.fallback=lucida,chinese-tw-iso10646,chinese-cn-iso10646,japanese-iso10646,korean-iso10646

# Exclusion Ranges

exclusion.japanese-iso10646=2200-22ef,2701-27be,20a0-20aa,2153-215f,2166-2168,216a-216f,2173-2182

# Font File Names

filename.-misc-ar_pl_shanheisun_uni-medium-r-normal--*-%d-*-*-c-*-iso10646-
1=/usr/share/fonts/chinese/TrueType/uming.ttf
filename.-misc-baekmuk_batang-medium-r-normal--*-%d-*-*-c-*-iso10646-1=/usr/share/fonts/korean/TrueType/batang.ttf
filename.-misc-baekmuk_gulim-medium-r-normal--*-%d-*-*-c-*-iso10646-1=/usr/share/fonts/korean/TrueType/gulim.ttf
filename.-misc-sazanami_gothic-medium-r-normal--*-%d-*-*-c-*-iso10646-
1=/usr/share/fonts/japanese/TrueType/sazanami-gothic.ttf
filename.-misc-sazanami_mincho-medium-r-normal--*-%d-*-*-c-*-iso10646-
1=/usr/share/fonts/japanese/TrueType/sazanami-mincho.ttf
filename.-misc-zysong18030-medium-r-normal--*-%d-*-*-c-*-iso10646-1=/usr/share/fonts/chinese/TrueType/zysong.ttf
filename.-b&h-lucidasans-medium-r-normal-sans-*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaSansRegular.ttf
filename.-b&h-lucidabright-medium-r-normal--*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaBrightRegular.ttf
filename.-b&h-lucidabright-demibold-r-normal--*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaBrightDemiBold.ttf
filename.-b&h-lucidabright-medium-i-normal--*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaBrightItalic.ttf
filename.-b&h-lucidabright-demibold-i-normal--*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaBrightDemiItalic.ttf
filename.-b&h-lucidasans-bold-r-normal-sans-*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaSansDemiBold.ttf
filename.-b&h-lucidasans-medium-i-normal-sans-*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaSansRegular.ttf
filename.-b&h-lucidasans-bold-i-normal-sans-*-%d-*-*-p-*-iso8859-1=$JRE_LIB_FONTS/LucidaSansDemiBold.ttf
filename.-b&h-lucidatypewriter-medium-r-normal-sans-*-%d-*-*-m-*-iso8859-
1=$JRE_LIB_FONTS/LucidaTypewriterRegular.ttf
filename.-b&h-lucidatypewriter-bold-r-normal-sans-*-%d-*-*-m-*-iso8859-1=$JRE_LIB_FONTS/LucidaTypewriterBold.ttf
filename.-b&h-lucidatypewriter-medium-i-normal-sans-*-%d-*-*-m-*-iso8859-
1=$JRE_LIB_FONTS/LucidaTypewriterRegular.ttf
filename.-b&h-lucidatypewriter-bold-i-normal-sans-*-%d-*-*-m-*-iso8859-1=$JRE_LIB_FONTS/LucidaTypewriterBold.ttf

# AWT X11 font paths
awtfontpath.chinese-tw-iso10646=/usr/share/fonts/chinese/TrueType
awtfontpath.chinese-cn-iso10646=/usr/share/fonts/chinese/TrueType
awtfontpath.japanese-iso10646=/usr/share/fonts/japanese/TrueType
awtfontpath.korean-iso10646=/usr/share/fonts/korean/TrueType
```

# Glossary

**American Standard Code for Information Interchange**
*a 7-bit encoding standard that provides a basic set of 128 characters, supporting a variety of computer systems. ASCII encodes the uppercase and lowercase letters of the English alphabet, punctuation marks, the digits 0–9, and control characters. This set of 128 characters is also included in most other encodings. Short form: ASCII.*

***ASCII***
*See American Standard Code for Information Interchange.*

**CEDA**
*See Cross-Environment Data Access.*

**Chinese, Japanese, and Korean**
*the collective group of languages Chinese, Japanese, and Korean. Short form: CJK.*

**CJK**
See Chinese, Japanese, and Korean.

**coded character set**
*a map of an abstract character repertoire to a set of numeric values. The ISO Latin-1 coded character set provides the Western European alphabet and symbols and their numeric representations. For example, the letter Ä is represented as C4 (hexadecimal). Short forms: CCS, character set.*

**Cross-Environment Data Access**
*a feature of SAS software that enables a SAS data file that was created in any directory-based operating environment (for example, Solaris, Windows, HP-UX, OpenVMS) to be read by a SAS session that is running in another environment. You can access the SAS data files without using any intermediate conversion steps. Short form: CEDA.*

**DATA step**
*in a SAS program, a group of statements that begins with a DATA statement and that ends with either a RUN statement, another DATA statement, a PROC statement, or the end of the job. The DATA step enables you to read raw data or other SAS data sets and to create SAS data sets.*

**DBCS**
*See double-byte character set.*

**double-byte character set**
*a character set that requires a variable-width encoding because many characters occupy two bytes of memory. The term DBCS, as traditionally applied to languages such as Japanese, Korean, and Chinese, is somewhat misleading because some DBCS characters actually require less than two bytes. Short form: DBCS.*

**EBCDIC**
*See Extended Binary Coded Decimal Interchange Code.*

**encoding**
*a mapping of a coded character set to code values*.

**Extended Binary Coded Decimal Interchange Code**
*a family of single-byte and multi-byte encodings for the representation of data on IBM mainframe and mid-range computers. EBCDIC encodes the uppercase and lowercase letters of the English alphabet, punctuation marks, the digits 0–9, and an extended set of control characters. Short form: EBCDIC.*

**font**
*a typeface with a specific character shape, spacing, weight, and size. The characters in a font can be figures, symbols, or alphanumeric*.

**glyph**
*the most basic element (a grapheme or combination of graphemes) of a typeface or font that is used to render text in a writing system.*

**I18N**
*See internationalization*.

**Internationalization**
*the process of designing a software product without making assumptions that are based on a single language or locale. Internationalization ensures that international conventions (including rules for sorting strings and for formatting dates, times, numbers, and currencies) are supported. It also facilitates a consistent user experience across different language editions of a product. Short form: I18N.*

**L10N**
*See localization.*

**legacy encoding**
*a single-byte  or multi-byte encoding pre-dating the Unicode standard. Legacy encodings are limited to the characters from a single language or a group of languages.*

**locale**
*a setting that reflects the language, local conventions, and culture for a geographic region. Local conventions can include specific formatting rules for paper sizes, dates, times, and numbers, and a currency symbol for the country or region. Some examples of locale values are French_Canada, Portuguese_Brazil, and Chinese_Singapore.*

**localization**
*the process of adapting software for a particular geo-cultural region (locale). Translation of the user interface, system messages, and documentation is a large part of the localization process. Short form: L10N.*

**multi-byte character set**
*a character set that requires a variable-width encoding because that characters occupies more than one byte of memory. DBCS and MBCS are sometimes used interchangeably, but MBCS is more accurate for describing the character sets of languages such as Japanese, Korean, and Chinese. Short form: MBCS.*

**multilingual**
*software that supports more than one natural language simultaneously, enabling the user to access content in multiple languages, but usually from a particular locale.*

**RDBMS**
*See relational database management system.*

**relational database management system**
*a database management system that organizes and accesses data according to relationships between data items. The main characteristic of a relational database management system is the two-dimensional table. Examples of relational database management systems are DB2, Oracle, Sybase, and Microsoft SQL Server. Short form: RDBMS.*

**SAS data set**
*a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. SAS data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data sets or from files whose contents are in other software vendors' file formats.*

**SBCS**
*See single-byte character set.*

**single-byte character set**
*a type of encoding for which each character is represented using one byte of computer memory. An example of a single-byte character set is ISO Latin 1. Short form: SBCS.*

**Unicode**
*a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems. Unicode includes more than 110,000 characters covering dozens of scripts, plus standards for character properties such as uppercase and lowercase, for rendering bidirectional script, and a number of related items.*

**Unicode server**
*a workspace server that is configured in SAS to handle Unicode*.

**Unicode Transformation Format 8**
*See UTF-8.*

**Unicode Transformation Format 16**
*See UTF-16*.

**Unicode Transformation Format 32**
*See UTF-32*.

**UTF-8**
*a multi-byte encoding that represents each Unicode character with 1 to 4 bytes. It is backward-compatible with ASCII.*

**UTF-16**
*a multi-byte encoding that represents each Unicode character with 2 or 4 bytes. It is not backward-compatible with ASCII*.

**UTF-32**
*a multi-byte encoding that represents each Unicode character with 4 bytes. It is not backward-compatible with ASCII.*

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2013 SAS Institute Inc., Cary, NC, USA. All rights reserved.