

Paper 025-2013

Internationalization 101: Give Some International Flavor to Your SAS® Applications

Mickaël Bouedo, SAS Institute Inc., Cary, NC, USA

Stephen Beatrous, SAS Institute Inc., Cary, NC, USA

ABSTRACT

Do you have SAS® users worldwide? Do you want your SAS application to be usable in many languages? SAS 9.4 internationalization features will get you there efficiently. If you want to adapt your SAS application for different cultures, SAS internationalization is the step that generalizes your product to be language independent. Internationalization features in SAS include the ENCODING, LOCALE, and TIMEZONE options; the SASMSG function; the NL formats; and many more features that help you write code once, so it can run in different cultural environments with low maintenance. This paper describes how to successfully internationalize your SAS programs and make them ready for the world.

INTRODUCTION: I18N OR L10N

Your customers and employees today likely speak many different languages. Why should your SAS applications targeted to those customers and employees only speak one language? Preparing your applications for a global audience requires the customization of your software for each target market. Typically, users want the user interface to appear in a language they understand. It should match the user's cultural characteristics, taking into account the complexity of their scripts as well as local conventions to format numbers, currency, dates, and times. Fonts, images, colors, addresses, page layout, and sorting are among other cultural differences to keep in mind.

The problem is how do you create an application for multiple languages that you might not speak? The answer is through globalization technologies.

Software Globalization, also abbreviated as "G11N", is the combined work of internationalization and localization. Software Globalization designs software for global markets, and then adapts the internationalized software to a specific locale.

Internationalization is the process of designing an application so that it can be adapted to various languages and regions without engineering changes. The term *internationalization* is often abbreviated as "I18N", because there are 18 letters between the first "i" and the last "n."

While the internationalization process prepares your application for the world, the localization process adapts it to the world. Localization is the process of modifying products or services to account for differences in distinct markets. With regard to software, it means adapting the software for a particular geographical region or for a set of users with common language and locale-specific customs. The term *localization* is often abbreviated as "L10N", because there are 10 letters between the "l" and the "n."

One of the primary tasks of localization is translating the user interface elements that have been externalized in the internationalization process. Localization involves changing not only the language interaction, but also other relevant changes such as the display of numbers, dates, currency, and so on. Other types of data, such as sounds and images, might require localization if they are culturally sensitive.

According to Microsoft documentation, an internationalized program has the following characteristics:

- World readiness: The ability for the application to support the customer's data and to format and sort according to the customer's preferences.

The first four sections of this paper are related to this category. The first section describes essential SAS options to the I18N process. The second section covers international formatting, while the next two chapters

introduce DATA step functions to support international data and linguistic sorting.

- **Localizability:** The act of preparing the software so that localization is possible. This includes externalizing text strings from source code and avoiding truncation of strings in the UI.

The last two sections of this paper cover this category as it introduces the language switching feature and a DATA step function to help with string externalization.

Without a proper and well-designed internationalization process, localization is not possible without re-writing the application for each language/culture. The better internationalized an application is, the easier it is to localize it for a particular language and character encoding scheme. With this document, you will know what to do to make your SAS programs ready for the local languages spoken in different countries. You will also learn the various ways to accomplish this.

INTERNATIONALIZATION CONCEPTS WITH THE SAS SYSTEM

In order to write applications for international users, it becomes critical to design applications that interact with users in their native language using local conventions. However, when any application is being developed, internationalization might not be given very much importance, as the focus is more to have the application working first and then maybe later on supporting different locales. Though this is not recommended, it might still work as long as the application is small and can take on the changes that follow after deployment for a particular (default) locale. But what if the application is more complex, with a large number of SAS programs, reports, or graphs, and the requirement is to support multiple locales? What factors should you consider before taking the actual step of changing your code to do this?

First, identify the areas in your application that will have to change in order to support users from different countries. There are two main areas that you will need to consider:

- The invisible parts of your application – support for different character encoding formats and the back-end logic that processes this data.
- The visible part of your application – the user interface, which includes the reports or graphs you will create. In SAS, the user interface specific changes could mean changes to text such as titles, footnotes, date formats, currency formats, etc.

For each area, the SAS System supports different options that will help you to archive your goal. For the first part about processing the data, the main option is the ENCODING option, which will set your session encoding. The encoding defines the character sets the SAS System uses to process syntax and user's data. The next section in this document describes in more detail the concept of session encoding and the importance of this option.

The second critical option to consider is the LOCALE option. This option will be the link between the user and your application. It allows the application to know the user's cultural preferences and will drive most of the user interface changes. The section "What is a Locale?" will tell you more about this option and how you could provide your own cultural data with the LOCALEDATA option and procedure.

A third option to consider when designing your application is the TIMEZONE option. This option, introduced in Release 9.4, can be used to set your time zone. It could be a helpful option especially when you have a unique SAS server that is running SAS programs from users located in different time zones, or when you have to process date or time data where geo-location is critical to your application.

1. CONCEPT OF SESSION ENCODING

An *encoding* is a set of abstract characters (letters, East Asian logograms, digits, punctuation marks, symbols, and control characters) that have been mapped to hexadecimal values (called *code points*) that can be used by computers. For example, IBM-1047 is an encoding in the EBCDIC family used mainly on the mainframe; Windows Cyrillic is an encoding in the Windows family; and Latin1 is an encoding in the ISO 8859 family.

Encodings have many properties. One possible categorization would be as follows:

- Single-byte character set (SBCS) encodings, which represent each character in a single byte. Each SBCS encoding supports 256 code points, which limits the number of characters available in the encoding. Some code points are reserved to interpret the control characters from the system. This number is enough to support languages based on the Latin script like most European languages, though even within Europe, several encodings are needed in order to cater to the character sets of the different languages. However, this is definitely not enough for other languages with a more complex script, such as the Chinese language or the Japanese language.
- Multi-byte character set (MBCS) encodings require one or more bytes to represent each character. Typically, a set of code points is reserved to represent the first byte of the set and is meaningful only if it is immediately followed by a defined second, third, or fourth byte. One of the most common MBCS encodings is UTF-8, where a character can be represented with up to 4 bytes. The double-byte character set (DBCS) is a form of MBCS.

MBCS encoded data can offer challenges for SAS programmers. When programming and manipulating MBCS data, you have to write code that would treat a set of one or more code points as one character. The section “Text String and Character Manipulation” explains how SAS can help you process data in such encodings.

The ENCODING option is used to specify what encoding to use to process data internally. The option value is also referred to as the *SAS Session encoding*. This option can be set at SAS invocation only, and it cannot be changed once SAS is started.

The languages you plan to support in your application determine the encoding value to select. SAS expects the data to be encoded in the encoding specified by the ENCODING option. Data sets have encoding properties that are used to transcode the data to the session encoding. For external sources of data such as flat files or other external databases, the encoding should be specified when accessing the data. However, the session encoding must support the characters you plan to process; otherwise, you might run the risk of data loss or corruption. For example, if your SAS application has to process Japanese, French, and English characters, then your session encoding will most likely be “UTF8”, which is an encoding that supports all characters from these three languages.

Table 1 lists three of the many encodings supported by the SAS System:

SAS encoding name	Description
-encoding LATIN1	Latin1 character encoding for languages in the Americas and Western Europe.
-encoding Shift-JIS	Character encoding for the Japanese language.
-encoding UTF-8	Character encoding for any language.

Table 1. Example of SAS encoding values

Refer to “Encoding Values for a SAS Session” in the *SAS 9.4® National Language Support (NLS): Reference Guide* for a complete list of encoding values.

2. WHAT IS A LOCALE?

A locale is a set of conventions for handling written language text (for example, word delimitation, quoting, or sorting) and various units (for example, numeric value, date and time formatting conventions, or currency conventions).

A locale identifies a specific user preference, or a group of users who have similar cultural and linguistic expectations for human-computer interaction (and the type of data they process). A locale’s identifier is a label for a given group of settings. For example, “en” (representing the English language) and “fr” (representing the French language) are identifiers for a cultural locale.

A locale represents a language in a geographical region. More than one locale can be associated with a particular language, which allows for regional differences. For example, an English-speaking user in the United States will use “en_US” (English language spoken in the United States) as the locale identifier, while an English-speaking user in Great Britain will use “en_GB” (English spoken in Great Britain).

Setting the LOCALE Option

In SAS, the LOCALE= system option must be used to specify the locale of the SAS session. The option can be set at SAS invocation and can be changed during the session. SAS supports two notations, a long one in the form

Language_territory and a short one in the form ll_rr, where ll represents the two-character language code and rr represents the two-character region code.

LOCALE option settings – short name	LOCALE option settings – long name
<pre>Option Locale = en_US ; Option locale = fr_CA ;</pre>	<pre>Option locale = English_UnitedStates ; Option locale = French_Canada ;</pre>

Table 2. Example of LOCALE option values

Refer to “Values for the LOCALE= System Option” in the *SAS 9.4® National Language Support (NLS): Reference Guide* for a complete list of locale values supported by the SAS System.

The setting of the LOCALE option affects the following SAS components:

- All NL formats and informats. The NL formats and informats are locale-sensitive formats/informats, which are described in the section “Locale-Sensitive Formatting”.
- SASMSG() and SASMSGF() functions used in the string externalization mechanism.
- Other options such as DFLANG, DATESTYLE, PAPERSIZE, MAPEBCDIC2ASCII, ODSLNGCHG, LOGLANGCHG, LSWLANG, LOCALEDATA, and TIMEZONE.

Modifying the Locale Data

When setting the LOCALE option to a specific value, the SAS System provides access to a various number of cultural data for the given locale value specified. These data are consumed by different SAS components such as the NL formats and informats. It is possible to modify these data with the LOCALEDATA procedure. This procedure enables you to manipulate the LOCALE data where you can view, open, and modify them. You can store your own customized locale data.

In order to be able to modify the LOCALE data, the SAS System must be started with the option LOCALEDATA set to “registry”.

In the SAS program below, the LOCALEDATA procedure modifies the key DATE_YEAR_FORMAT, which is used by the NLDATYRw. format. The updated format will now show the value with the string “Year” followed by the numeric year between parentheses.

SAS code	Result
<pre>/* Start SAS with option -LOCALEDATA registry */ PROC LOCALEDATA; LOAD SASLOCALE; MODIFY key=DATE_YEAR_FORMAT value='Year(%Y)' ; SAVE REGISTRY / _ALL_ syntax=SAS; data _null_ ; d = today() ; put d nldateyr. ; run ;</pre>	<p>Year(2013)</p>

Code Box 1. Example of the LOCALEDATA procedure

There are many keys used mainly by the NL formats/informats that can be modified. For a complete list of the keys, refer to “LOCALEDATA Procedure” in the *SAS 9.4® National Language Support (NLS): Reference Guide*.

3. WHAT IS YOUR TIMEZONE?

In Release 9.4, the SAS System supports a new option called TIMEZONE. This option allows a user to specify a time zone ID or a time zone name in order for SAS to use a particular time zone.

The TIMEZONE option supports different notations:

SAS Time Zone	Description
<code>Option timezone = 'America/Denver' ;</code>	Time zone ID. This is the recommended notation because this value contains the Daylight Saving Time information.
<code>Option timezone = 'PST8' ;</code>	Standard time zone name
<code>Option timezone = 'SASNC-2:30' ;</code>	User-defined time zone name

Table 3. Example of SAS TIMEZONE values

Refer to “TIMEZONE Values for a SAS Session” in the *SAS 9.4® National Language Support (NLS): Reference Guide* for complete list of TIMEZONE values.

The time zone setting affects the following SAS components:

- Times that are recorded by events and logs.
- Time of data set creation or modification.
- Date and Datetime functions: DATE(), DATETIME(), TIME(), TODAY()
- Time zone functions: TZONEOFF(), TZONEID(), TZONENAME(), TZONES2U(), TZONEU2S(), TZONESTTNAME(), TZONEDSTNAME(), TZONESTTOFF(), TZONEDSTOFF()
- Time zone formats: B8601DXw, E8601DXw., B8601LXw., E8601LXw., B8601TXw., E8601TXw., NLDATMzw., NLDATMTzw., and NLDATMWzw.

LOCALE-SENSITIVE FORMATTING

Around the world, people of all cultures encounter different ways of writing dates, times, and numbers. Though most cultures convey the same information, there are differences in the way this data is formatted. When you display data to a user in an international context, it should be formatted according to the conventions of the user’s native country, region, or culture. When users enter data, they might do so according to their own customs or preferences. For example, in the United States the string “15.125” represents the decimal number fifteen and one eighth, whereas in Germany it represents fifteen thousand, one hundred and twenty-five.

SAS provides support to read and write such locale-sensitive data. SAS language elements called National Language Support (NLS) formats and informats enable SAS to convert date, datetime, currency, and numeric values back and forth in a locale-sensitive manner.

Informats read notations or expressions, such as a number, a clock time, or a calendar date (which might be in a variety of lengths based on the culture), and then convert the data to a SAS value. Formats write a value, recognized by SAS, such as a time or date value, as a calendar date or clock time in a variety of lengths and notations according to the SAS locale session.

1. DATE

Cultures around the world have different ways of representing dates. Date formats do not necessarily rely on language; speakers using the same language from different countries might format dates differently. Writing a date requires placing the date’s elements (for example, the month, the day, and the year) in a way that is understandable to the reader.

For example, what date represents “04/02/12”? This date is interpreted differently depending on your cultural background. In the United States, it represents “02 April, 2012”. In Japan, it is interpreted as “12 February, 2004”. And in most parts of Europe, it means “04 February, 2012”.

SAS has several formats and informats listed in the table below to process date values in a locale-sensitive manner.

Name	Description	Option locale=en_US ;	Option locale=de_DE;
NLDATE	Date	January 22, 2012	22. Januar 2012
(9.4) NLDATEL	Date Long	January 22, 2012	22. Januar 2012
(9.4) NLDATEM	Date Medium	Jan 22, 2012	22.01.2012

(9.4) NLDATES	Date Short	01/22/2012	22.01.2012
NLDATEMD	Day and name of the month	January 22	22 Januar
(9.4) NLDATEMDL	Day & name of the month Long	January 22	22 Januar
(9.4) NLDATEMDM	Day & name of the month Medium	Jan 22	22 Jan
(9.4) NLDATEMDS	Day & name of the month Short	01/22	22.01
NLDATEMN	Name of the month	January	Januar
NLDATEW	Day and date of the week	Sunday, January 22, 2012	Sonntag, 22. Januar 2012
NLDATEWN	Day of the week	Sunday	Sonntag
NLDATEYM	Year and name of the month	January 2012	Januar 2012
(9.4) NLDATEYML	Year & name of the month Long	January 2012	Januar 2012
(9.4) NLDATEYMM	Year & name of the month Medium	Jan 2012	Jan 2012
(9.4) NLDATEYMS	Year & name of the month Short	01/2012	01.2012
NLDATEYQ	Year and quarter	1st quarter 2012	1. Quartal 2012
(9.4) NLDATEYQL	Year and quarter	1st quarter 2012	1. Quartal 2012
(9.4) NLDATEYQM	Year and quarter	Q1 2012	Q1 2012
(9.4) NLDATEYQS	Year and quarter	2012/1	2012.1
NLDATEYR	Year	2012	2012
NLDATEYW	Year and week	Week 04 2012	Week 03 2012

Table 4. List of the NL Formats and NL Informats Available for a Date Value

2. TIME AND DATETIME

Along with date formats, time and datetime formats differ culturally. Time formats are usually divided between 12-hour formats and 24-hour formats. For example, in the United States, the 12-hour time notation is the most common way to present the time. Most other countries in the world use a 24-hour time format, also known as “military time” in the United States.

Name	Description	Option locale=en_US ;	Option locale=fr_FR;
NLTIME	Time	23:52:12	23 h 52
NLTIMAP	Time value with a.m. or p.m.	11:52 PM	23 h 52
NLDATM	Date and time.	22Jan12:23:52:12	22 janvier 2012 23 h 52
(9.4) NLDATML	Date and time Long	January 22, 2012 11:52:12 PM	22 janvier 2012 23:52:12
(9.4) NLDATMM	Date and time Medium	Jan 22, 2012 11:52:12 PM	22 janv. 2012 23:52:12
(9.4) NLDATMS	Date and time Short	01/22/2012 23:52:12	22/01/2012 23:52:12
NLDATMAP	Date and time with a.m. or p.m.	January 22, 2012 11:52:12 PM	22 janvier 2012 23 h 52
NLDATMDT	Date	January 22, 2012	22 janvier 2012
NLDATMMD	Name and day of the month	January 22	22 janvier
(9.4) NLDATMMDL	Name & day of the month Long	January 22	22 janvier
(9.4) NLDATMMDM	Name & day of the month Medium	Jan 22	22 janv.
(9.4) NLDATMMDS	Name & day of the month	01/22	22/01

	Short		
NLDATMMN	Name of the month	January	Janvier
NLDATMTM	Time of the day	23:46:12	23 h 52
NLDATMW	Day of the week and datetime	Sunday, Jan 22, 2012 11:52:12 PM	dimanche 22 janvier 2012 23 h 52
NLDATMWN	Day of the week	Sunday	Dimanche
NLDATMWZ	Day, datetime, and time zone	Sun, Jan 22, 2012 11:52:12 PM -0500	Dim. 22 janvier 2012 23 h 52 -0500
NLDATMYM	Year and name of the month	January 2012	janvier 2012
(9.4) NLDATMYML	Year & name of the month Long	January 2012	janvier 2012
(9.4) NLDATMYMM	Year & name of the month Medium	Jan 2012	janv. 2012
(9.4) NLDATMYMS	Year & name of the month Short	01/2012	01/2012
NLDATMYQ	Year and quarter of the year	1st quarter 2012	1er trimestre 2012
(9.4) NLDATMYQL	Year and quarter Long	1st quarter 2012	1er trimestre 2012
(9.4) NLDATMYQM	Year and quarter Medium.	Q1 2012	T1 2012
(9.4) NLDATMYQS	Year and quarter of the Short	2012/1	2012/1
NLDATMYR	Year	2012	2012
NLDATMYW	Year and name of the week	Week 04 2012	Week 03 2012
NLDATMZ	Time zone and datetime	22Jan12:23:52:12 -0500	22 janvier 2012 23 h 52 -0500

Table 5. List of the NL Formats and NL Informat Available for a Datetime Value

2.1 Time Zone Formats and Informat

In Release 9.4, the SAS System supports new formats and inmat related to the TIMEZONE option, especially formats and inmat to support ISO 8601 standard. International standard ISO 8601 specifies numeric representations of date and time to cover the exchange of date- and time-related data. This standard notation is to provide a clear and well-defined method of representing dates and times, and to avoid misinterpretation of numeric representations of dates and times, particularly when data is transferred between countries with different conventions for writing numeric dates and times.

In addition to the support for ISO 8601, three new NL formats were also added to provide locale-sensitive notation in addition to the time zone support.

In the table below, the examples use the new formats by formatting the following date and time:

```
datetime = '29apr2013:16:11:45'dt  time = '16:11:45't
```

Name	Description	TZ='America/Denver' locale=en_US	TZ='Europe/Paris' locale=fr_FR
B8601DX	Converts UTC datetime value into user local time and writes local time (SAS datetime value) with time zone offset.	20130429T101145-0600	20130429T181145+0200
B8601LX	Writes SAS datetime value with timezone offset using the ISO 8601	20130429T161145-0600	20130429T161145+0200

	basic notation.		
B8601TX	Converts UTC time into user local time and writes local time value with timezone offset.	091145-0700	171145+0100
E8601DX	Converts UTC datetime value into user local time and writes local time (SAS datetime value) with time zone offsets.	2013-04-29T10:11:45-06:00	2013-04-29T18:11:45+02:00
E8601LX	Writes SAS datetime value with timezone offsets.	2013-04-29T16:11:45-06:00	2013-04-29T16:11:45+02:00
E8601TX	Converts UTC time into user local time and writes local time value with timezone offsets.	09:11:45-07:00	17:11:45+01:00
NLDATMZ	Converts SAS datetime values to the locale-sensitive datetime string as time zone and datetime.	29Apr2013:16:11:45 -0600	29 avril 2013 16 h 11 +0200
NLDATMTZ	Converts the time portion of the SAS datetime of the locale to the time of day and time zone.	16:11:45 -0600	16 h 11 +0200
NLDATMWZ	Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.	Monday, April 29, 2013 04:11:45 PM -0600	lundi 29 avril 2013 16:11:45 +0200

Table 6. List of the Formats and Informats Related to the TIMEZONE Option

3. NUMBERS

In an international environment, formatting numbers properly and uniformly is essential to avoid miscommunication. The most common cultural differences regarding the number formatting are decimals group separators and negative representation.

In the United States, a comma separates the thousands and a period separates a whole number from a decimal value. In some countries, like Germany, the two punctuation marks have reversed roles. In other countries, different characters are used.

Name	Description	Option locale=en_US;	Option locale=ru_RU;
NLBEST	The best numerical notation.	-123456.789	-123456,789
NLNUM	numeric format of the local expression.	-123,456.79	-123 456,79
NLNUMI	numeric format of the international expression.	-123,456.79	-123,456.79
NLPCT	percentage data of the local expression.	56.84%	56,84%
NLPCTI	percentage data of the international expression.	56.84%	56.84%
NLPCTN	percentages, using a minus sign for negative values.	56.84%	56,84%
NLPCTP	locale-specific numeric values as percentages.	56.84%	56,84%
NLPVALUE	p-values of the local expression.	0.57	0,57

Table 7. List of the NL Formats and NL Informats Available for Numbers

4.CURRENCY

Formatting currency usually follows the same rules as formatting numbers; however, the position of the currency symbol varies widely from country to country.

Name	Description	Option locale=en_US ;	Option locale=es_ES;
NLMNY	monetary format of the local expression.	(\$127,654.30)	-127.654,30€
NLMNYI	monetary format of the international expression.	(USD127,654.30)	-127.654,30EUR

Table 8. List of the NL Formats and NL Informats Available for Currency

The formats and informats listed here only change currency symbols and patterns based on the locale and simply display the number found in the data. **NO currency conversion is performed.**

To cover the case where the currency is fixed but the format is locale-sensitive, SAS supports additional NLS formats and informats. They use the following naming convention: NLMNLxxx or NLMNIxxx, where xxx is the international code for a given currency. About 37 currencies are supported through these formats. The table below lists some of these formats and informats.

Name	Description	Option locale=en_US ;	Option locale=es_ES;
NLMNLEUR	Writes the monetary format of the local expression for the Euro currency.	(€12,358.00)	-12.358,00€
NLMNIEUR	Writes the monetary format of the international expression for the Euro currency.	(EUR12,358.00)	-12.358,00EUR
NLMNLCNY	Writes the monetary format of the local expression with the Chinese currency.	(RMB12,358.00)	-12.358,00RMB
NLMNICNY	Writes the monetary format of the international expression with the Chinese currency.	(CNY12,358.00)	-12.358,00CNY
NLMNY	monetary format of the local expression.	(\$127,654.30)	-127.654,30€
NLMNYI	monetary format of the international expression.	(USD127,654.30)	-127.654,30EUR

Table 9. Additional NL Formats and NL Informat Available for Currency

5. CUSTOM FORMATS

The FORMAT procedure with the LOCALE statement allows you to create a library of custom formats for any languages supported by the SAS System. The procedure creates a catalog that corresponds to the current SAS locale. The name of the catalog that SAS creates is the SAS library or catalog that is specified in the LIBRARY= option appended with the five-character POSIX locale value for the current SAS locale. To access the formats based on the LOCALE option, the option FMTSEARCH must be set with the statement “/locale”.

In the example below, the SAS program creates two catalogs of formats in the WORK library. The first PROC FORMAT instruction creates the default catalog called *work.formats*, while the second PROC FORMAT instruction creates the catalog *work.formats_fr_ca*.

When formatting the data, the setting of the LOCALE option determines which catalog and formats to use. If the LOCALE is set to “fr_CA”, then SAS uses the formats from the catalog *work.formats_fr_ca*. For any other locales, SAS uses the formats from the default catalog.

```

/*no locale information*/
proc format lib=work.formats;
value size low - 56 = 'Short'
           57 - 66 = 'Medium'
           20 - high = 'Tall' ;
run;
options locale=fr_CA;
proc format lib=work.formats locale;
value size low - 10 = 'Petit'
           11 - 19 = 'Moyen'
           20 - high = 'Grand' ;
run;

```

```

options fmtsearch=(work/locale);

/* Use the English format catalog*/
title "Locale is English, Use the Default Format Catalog";
proc print data=sashelp.class;
format height size. ;
run;
/* Use the Romanian format catalog*/
options locale=fr_CA;
title 'Locale is fr_CA, Use the French Format Catalog';
proc print data= sashelp.class;
format height size. ;
run;

```

Code Box 2. Example of the FORMAT Procedure with the Locale Option

TEXT STRING AND CHARACTER MANIPULATION

SAS 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 character set (SBCS). However, multi-byte character sets (DBCS or MBCS) require one byte or more to represent the characters in the set. Using the original SAS string-handling functions with multi-byte data can lead to unexpected behavior, such as data truncation.

To resolve issues that 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 SBCS data, because one character is always 1 byte in length, you can assume that the second character in the string begins in byte 2 of the string. However, if the data in the string is multi-byte data, the data in byte 2 might be one of the following, depending on the data and the 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 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 SBCS, DBCS, and MBCS (UTF-8) data in SAS.

A typical problem

Consider the following string, which is represented by its hexadecimal value:

```
STR=' 8361836262'x
```

In a double-byte encoding such as Shift-JIS, this string has 5 bytes and represents three characters. In a single-byte encoding such as LATIN1, this same string represents something entirely different. It represents five characters as shown in Figure 1.

Shift-JIS Characters	ヂ		ツ		b
Hexadecimal Representation	83	61	83	62	62
Latin1 Characters	f	a	f	b	b

Figure 1. String representation in Latin1 and Shift-JIS encoding

The following example uses this same hexadecimal string in the INDEX functions, which returns a value of 4 for both the SBCS and the multi-byte environments. However, the result is incorrect for the multi-byte environment because the sequence of characters 'bb' does not exist in the string in a DBCS encoding. In a multi-byte environment, the result should be 0.

With the SCAN function, the expected returned string would be the first two Japanese characters. However, the function returns only the first one followed by another character that does not even seem to be in the string.

<pre>/* SAS program to submit in a MBCS SAS session */ data _null_ ; str= "ヂツb" ; i=index(str, "bb"); s=scan(str,1,'b') ; l=length(str) ; put str= \$hex10. /i= /s= l=; run ;</pre>	<pre>/* Results */ str=8361836262 i=4 s=ヂ l=5</pre>
--	---

Code box 3. Code example using regular functions

Solution

To avoid such issues, you can substitute the INDEX() function, which only works on single-byte data, with the KINDEX() function, which works on single- and multi-byte data. Direct substitutions can also be made for other string-handling functions, such as changing the LENGTH() or SUBSTR() functions to use the KLENGTH() or KSUBSTR() functions instead. Table 10 "Differences between K Functions and Regular SAS Functions" provides a list of available K functions.

<pre>/* SAS program to submit in a MBCS SAS session */ data _null_ ; str= "ヂツb" ; i=kindex(str, "bb"); s=kscan(str,1,'b') ; l=klength(str) ; put str= \$hex10. /i= /s= /l=; run ;</pre>	<pre>/* Results */ str=8361836262 i=0 s=ヂツ l=3</pre>
--	--

Code box 4. Code example using K functions

Because the two functions read the data differently, you will get different results when running this code in a multi-byte environment. The INDEX function is byte-oriented, so it reads the individual bytes of data. The INDEX function finds the hexadecimal byte sequence '6262'x, which it incorrectly interprets as the sequence 'bb' and returns a value of 4. The KINDEX function, on the other hand, is character based and interprets the first '62'x as the second byte of the DBCS character rather than as an ASCII 'b'. Therefore, KINDEX returns 0 instead of 4. In an SBCS environment, the KINDEX function works as the INDEX function.

K Function Compatibility

The following table lists the original SAS string-handling functions and the equivalent K functions. Note that a few of the original string-handling functions support multi-byte data correctly and can continue to be used in a multi-byte environment. For example, UPCASE and LOWCASE operate correctly on multi-byte data.

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 there are some behavior differences. Table 10 only notes major differences, such as differences in the arguments. For details about

these differences refer to “Internationalization Compatibility for SAS String Functions” in the SAS 9.4® *National Language Support (NLS): Reference Guide*.

SAS Functions	Equivalent K Functions	Description	Is K Function Required for multi-byte characters? Comments	Is K function a direct replacement?
CAT	KSTRCAT	Concatenates character strings without removing leading or trailing blanks.	CAT cannot handle contiguous SO/SI characters properly under an EBCDIC environment	Yes
COMPARE	KCOMPARE	Returns the result of a comparison of character strings.	Yes	arguments are different
COMPRESS	KCOMPRESS	Removes specific characters from a character string.	Yes	arguments are different
	KCOUNT	Returns the number of multi-byte characters in a string. KCOUNT is not compatible with COUNT.		
	KCVT	Converts data from one type of encoding data to another type of encoding data.		
INDEX	KINDEX	Searches a character expression for a string of characters.	Yes	Yes
INDEXC	KINDEXC	Searches a character expression for specific characters.	Yes	Yes
LEFT	KLEFT	Left aligns a character expression by removing unnecessary leading DBCS blanks.	LEFT is acceptable if DBCS blanks are not used	Yes
LENGTHN	KLENGTH	Returns the length of an argument.	Yes	Yes
LOWCASE	KLOWCASE	Converts all letters in an argument to lowercase.	No	Yes
REVERSE	KREVERSE	Reverses a character expression.	Yes	Yes
RIGHT	KRIGHT	Right aligns a character expression by trimming trailing DBCS blanks.	RIGHT is acceptable if DBCS blanks are not used	Yes
SCAN	KSCAN	Selects a specified word from a character expression.	Yes	arguments are different
STRIP	KSTRIP (9.4)	Removes leading and trailing blanks from a character string	Yes	Yes
SUBSTR	KSUBSTR	Extracts a substring from an argument.	Yes	Yes
	KSUBSTRB	Extracts a substring from an argument according to the byte position of the substring in the argument.		
TRANSLATE	KTRANSLATE	Replaces specific characters in a character expression.	Yes	direct replacement but does not support binary data like TRANSLATE
TRIM	KTRIM	Removes trailing DBCS blanks from character expressions.	Yes	Yes
	KTRUNCATE	Truncates a character string value to a specified length.		
UPCASE	KUPCASE	Converts all single-byte letters in an argument to uppercase.	No	Yes
	KUPDATE	Inserts, deletes, and replaces the contents of the character value according to the character-based position of the character value in the argument.		
	KUPDATEB	Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.		
VERIFY	KVERIFY	Returns the position of the first character that is unique to an expression.	Yes	Yes

Table 10. Differences between K Functions and Regular SAS Functions

Recommendations

The K functions work in an SBCS environment. However, before replacing all of the original SAS string-handling functions with K functions, examine your SAS program. If the string function processes data that will contain only 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 is in your SAS programs and how it is processed can save unnecessary updates to your SAS code. The processing of binary data is not supported by the string-handling K functions, which expect strings to match the current session encoding.

COLLATION IN SAS

To sort data in a linguistically correct order, use PROC SORT with the option SORTSEQ=LINGUISTIC. This option causes the SORT procedure to use the default collation rules consistent with the current LOCALE= setting. Additional options can be specified to customize the collation order.

An example of a collating rule for linguistic collation:

```
/* Sorting based on the current SAS Locale session */  
PROC SORT data=mydata SORTSEQ=LINGUISTIC;  
/* Sorting for a specific locale */  
PROC SORT data=mydata SORTSEQ=LINGUISTIC (LOCALE=fr_FR);
```

Code box 5. Code example using the linguistic sort

More information about the SORT procedure is available in the papers "Creating Order out of Character Chaos" and "Linguistic Collation: Everyone Can Get What They Expect".

LANGUAGE SWITCHING

The language switching feature allows you to produce multilingual reports on the fly by switching languages during the same SAS session. This concept has been introduced in Release 9.3, and it has been enhanced in Release 9.4 where new options are available to control the language of the LOG and the OUTPUT.

This feature is particularly interesting with a Unicode server that can support many different languages. The language switching feature takes advantage of the existing translations provided by the SAS System 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.

The language used for messages displayed by SAS is determined at start up by setting appropriate options in the SAS configuration file. The language selected must be compatible with the session encoding. The feature, referred to as "Language Switching" is controlled by several options:

- **LOGLANGCHG** Option.
Introduced in SAS 9.4, this option enables the language of message switching in SAS log output. If LOGLANGCHG is specified, the language of SAS log depends on the 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, the LOCALE= option determines the language of SAS log.

- **ODSLANGCHG** Option.
Also new in SAS 9.4, this option enables the language of message switching in SAS ODS output. If ODSLNGCHG is ON, the language of ODS output depends on the 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, the LOCALE= option determines the language of ODS output.
- **LSWLANG** Option.
Another new option in SAS 9.4, LSWLANG specifies the language of messages if LOGLANGCHG or ODSLNGCHG is ON. The option accepts one of the following valid values:

Valid values for the LSWLANG option	
LOCALE	
EN	English
FR	French
IT	Italian
DE	German
ES	Spanish
ZH	ChineseS
ZT	ChineseT
JA	Japanese
KO	Korean
PL	Polish
RU	Russian

Table 11. Valid values for the LSWLANG option.

The default setting of LSWLANG is the string “LOCALE”, which means the LOCALE= option will determine the language for switching. The setting of the LSWLANG= option also needs to be compatible with SAS session encoding; otherwise, its value will be ignored.

- **LOGLANGENG** Option.
It is a new option in SAS 9.4. It is a toggle option. When LOGLANGENG is enabled, it overrides LOGLANGCHG and LSWLANG and sets them to LOGLANGCHG=ON, LSWLANG=“EN”. As result, the LOG output will be in English, and NL Format output has no change. This option affects the setting of ODSLNGCHG. If ODSLNGCHG=OFF, the language of MVA message for ODS output will be determined by SAS configuration. If ODSLNGCHG=ON, all messages will be in English because of the LSWLANG setting.
- **LOCALELANGCHG** Option.
The LOGCALELANGCHG= option was introduced in a previous SAS release, and it acts as an alias of ODSLNGCHG in SAS 9.4.

In order to use the language switching feature, see the SAS program example below. The SAS System has been started with the following options at invocation:

```
sas -lswlang locale -odslngchg -nologlangchg
```

The SAS program creates a report with basic statistics using an ODS template from the FREQ procedure. The same program is executed twice: once with the LOCALE option set to “en_US” and a second time with the option changed to “fr_FR”. An HTML output is created for each execution: one in English and the other one in French. The French translations are coming from the localized SAS components.

```

options locale=fr_FR; /* en_US */
title1 "LSWLANG=%SYSFUNC(getoption(LSWLANG))" ;
title2 "LOCALE=%SYSFUNC(getpxlocale()) " ;
footnote "%sysfunc(datetime(),nldatm.)";
proc freq data=sashelp.class;
  table age / measures ;
run;

```

Code box 6. Language switching feature

LSWLANG=LOCALE LOCALE=en_US					
The FREQ Procedure					
Age	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
11	2	10.53	2	10.53	
12	5	26.32	7	36.84	
13	3	15.79	10	52.63	
14	4	21.05	14	73.68	
15	4	21.05	18	94.74	
16	1	5.26	19	100.00	

21Feb2013:11:35:42

LSWLANG=LOCALE LOCALE=fr_FR					
Procédure FREQ					
Age	Fréquence	Pourcentage	Fréquence cumulée	Pctage cumulé	
11	2	10.53	2	10.53	
12	5	26.32	7	36.84	
13	3	15.79	10	52.63	
14	4	21.05	14	73.68	
15	4	21.05	18	94.74	
16	1	5.26	19	100.00	

21 février 2013 13 h 57

Figure 2. Resulting HTML output with the Language switching feature

STRING EXTERNALIZATION

Localization of a product requires translation of the text that appears to your customers. For products written in SAS, it means strings in the SAS code must be translated. One way to accomplish this would be to provide customized localizations of your application. A copy of each .sas file that contains hardcoded strings would be needed, and it would result in a separate version of your application for each language. This is a tedious approach that could produce results that are different from one language to another and would require high maintenance and testing costs. Every time a new feature is needed in your application, the feature must be added to each localized version. The alternative to customized localizations is the string externalization process, which is part of the internalization approach.

This section of the paper describes this simple process, which you can use to localize your own SAS programs. The process involves moving hardcoded strings from your SAS code to a special properties file, called an .smd file. After the strings are in the .smd file, a copy of the .smd file is created for each translation of your application. When translations are complete, you run a SAS macro (%smd2ds) that puts the original and translated messages into a SAS data set. Finally, you modify your code to call the SASMSG function to retrieve the strings properly.

STEP 1 – IDENTIFY AND MOVE TRANSLATABLE STRINGS INTO AN .SMD FILE

Here are some examples of text strings that are hardcoded in SAS code. As you can see, some of these strings contain macro variables that substitute additional text.

```

put "Welcome back &user!" ;
TITLE "Hello &user! Today's date is &today";
FOOTNOTE "Report generated on &SYSDATE" ;

```

```
%PUT The file "&aFile" does not exist. ;
attrib room label="Number of rooms";
proc format ; value answer 1="Yes" 2="No" ; run ;
```

Code box 7. Examples of embedded strings in SAS code

All text strings in your SAS programs must be removed and placed in a file with the .smd extension. A key, or unique name, must be associated with each translatable string moved to the .smd file. Each string and its key are stored as a pair of strings, often referred to as a *key/value pair* of strings.

The format to store the key/value pair uses the equal sign as a delimiter. The key name must contain 7-bits ASCII characters only, and the size is limited to 60 characters.

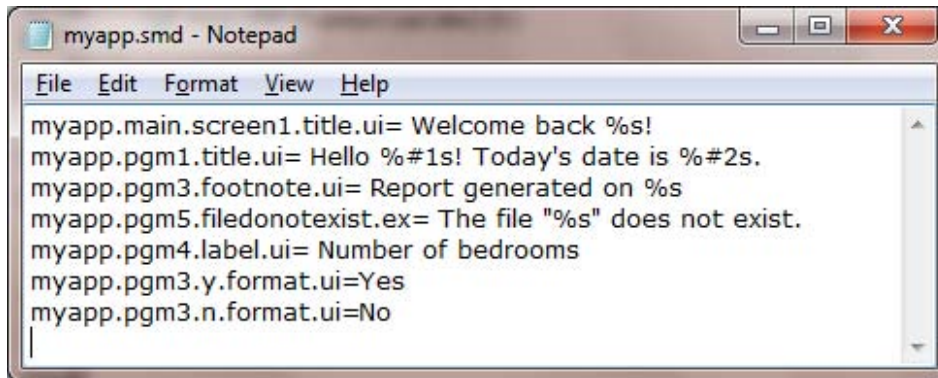


Figure 3. Example of an .smd file

Naming and Character Encoding of the .smd Files

The name of the .smd file is used as the name of a SAS data set. Any restriction that applies to a SAS data set name also applies to the name of the .smd file.

When you create a localized version of your .smd file, you use the name of the original .smd file and append an underscore (_) followed by the language or locale identifier. Therefore, do not use a name for your .smd file that ends with the underscore character _ followed by two letters because this ending is reserved for localized files.

For example, if you had created a file myapp.smd and the file is translated into Spanish for Mexico and French, you should have the following .smd files in your directory:

```
myapp.smd
myapp_es_MX.smd
myapp_fr.smd
```

The .smd files must be created in an encoding-neutral format called 7bits-ASCII. All characters that cannot be expressed as ASCII must be represented with the Unicode escape sequence, which uses the form \uxxxx, where xxxx is the hexadecimal representation of the Unicode character. For example, the character "small letter e with acute accent" (é) would be represented as \u00E9 in the .smd file.

A tool or utility can be used to convert the file to use Unicode escapes for non-ASCII characters. The Java tool native2ascii is a handy tool included in the JDK, which is used to convert a file with non-Latin 1 or non-Unicode characters to Unicode-encoded characters using the Unicode escape sequence.

SAS can also perform such conversions using the KPROPDATA function or the UNICODDEC function.

```
%MACRO SMD2ASCII (INF=,OUTF=,LRECL=,INENCODING=) ;
DATA _NULL_ ;
  attrib tmp length=$ &LRECL ;
  INFILE "&INF" lrecl=&LRECL ;
  INPUT ;
  FILE "&OUTF" lrecl=&LRECL ;
  tmp = kpropdata(_INFILE_, "UESC", "&INENCODING", "ASCII") ;
  put tmp ;
```



```

RUN ;
%MEND SMD2ASCII ;

%SMD2ASCII( INF = U:\tmp\myprod_pl_wlt2.smd,
            OUTF = U:\smd\ascii\myprod_pl.smd,
            INENCODING=wlatin2,
            LRECL=300 ) ;

```

Code Box 8. Example of a macro to convert an .smd file from a given encoding to ASCII with Unicode escape notation

Providing Context for Externalized Strings

One of the drawbacks to removing strings from SAS code is that the translators will not know the context in which the string was used. In order to provide useful information to help translators and the localization process, here are a few suggestions you can apply:

- Create your key names with a naming convention that gives a hint about how the text is used. For example, you could add a suffix to the key to categorize your strings:
 - my_key_title where the suffix "_title" indicates the string is used as a title.
 - my_key_footn for a string used as a footnote statement.
 - my_key_label for a string used as a label for a variable.
 - my_key_warning for a warning message.
- Append a number to the key name to indicate a size limit:
 - my_key_title_40
- Separate your messages into categories.
- Add comments for the context. A comment must begin with the # character.

Substitutions in the Message Text

Substitutions are made into message text as text strings. As many as seven substitutions can be made into each message. A substitution is represented in the .smd file using the %s formatting specifier.

```
DI_ATTR_WARN=The attribute "%s" is already registered for the table %s.
```

In the SAS code, the substitutions are passed to the SASMSG function as character strings.

During translation of an .smd file, the order of a substitution might change in the message text depending on the order of words for the language. SAS programmers or translators might add an argument number specification (%#1s %#2s ...) to indicate the order in which substitutions are to be made from the SASMSG argument list.

STEP 2 – CREATE A SAS DATA SET FROM THE .SMD FILE

The next step in the string externalization process is to create a SAS message data set from the .smd files. This step is done once by an administrator to gather the localizations into a single SAS data set. The SAS data set is the only run-time component of the application.

Some restrictions apply for the data set:

- The data set must contain the four variables described in Table 10.
- The SAS message data set must be a 7-bit ASCII data set with an encoding of US-ASCII or ASCIIANY.
- The data set must be sorted on the following variables: *locale*, *key*, and *lineno*, with the latter in descending order.
- A composite index on the variables *locale* and *key* must be defined.

#	Variable Name	Type	Length	Description
1	locale	char	5	language of the message
2	key	char	60	key to identify the message
3	lineno	num	5	line # of the message in reverse order
4	text	char	1200	1,200 text of the message

Table 12. Description of a SAS message data set

Such data sets can be created with the %SMD2DS() macro function. The macro %SMD2DS is available in the SAS AUTOCALL library:

```
%SMD2DS (DIR=, BASENAME=, LOCALE=, LIB=)
```

where

DIR= is the directory where the .smd files are located.

BASENAME= is the basename of the file to process.

LOCALE= <optional> is the list of locales to include separated by a blank. The default file to process is *basename.smd* if the parameter is not specified.

LIB= <optional> is the library where you create the data set. The default library is WORK if the parameter is not specified.

```
%SMD2DS (DIR=c:\myplaypen\myproduct\smd, BASENAME=myapp, LOCALE=fr ja_JP) ;
```

In the example here, the macro %SMD2DS looks for a file called myapp.smd in the directory specified within the DIR parameter. If the file does not exist, the macro looks for the file myapp_en.smd. If that file does not exist either, the macro stops. In addition to the default files, the macro looks for files that match the LOCALE= parameter. Here, the files are myapp_fr.smd and myapp_de_DE.smd. The data set is created in the WORK library.

The following example shows what a sasmsg data set looks like. In this data set, the messages have been translated into French and Japanese and converted to 7-bit ASCII characters.

	locale	key	lineno	text
4	en	MYAPP.PGM3.N.FORMAT.UI	1	No
5	en	MYAPP.PGM3.Y.FORMAT.UI	1	Yes
6	en	MYAPP.PGM4.LABEL.UI	1	Number of bedrooms
7	en	MYAPP.PGM5.FILEDONOTEXIST.EX	1	The file "%s" does not exist.
8	fr	MYAPP.MAIN.SCREEN1.TITLE.UI	1	Bienvenue %s!
9	fr	MYAPP.PGM1.TITLE.UI	1	Bonjour %s! Aujourd'hui nous sommes le %s.
10	fr	MYAPP.PGM3.FOOTNOTE.UI	1	Rapport glu00e9n\u00e9r\u00e9 le %s
11	fr	MYAPP.PGM3.N.FORMAT.UI	1	Non
12	fr	MYAPP.PGM3.Y.FORMAT.UI	1	Oui
13	fr	MYAPP.PGM4.LABEL.UI	1	Nombre de chambres
14	fr	MYAPP.PGM5.FILEDONOTEXIST.EX	1	Le fichier "%s" n'existe pas.
15	ja_JP	MYAPP.MAIN.SCREEN1.TITLE.UI	1	\u304a\u304b\u3048\u308a\u306a\u3055\u3044 %s!
16	ja_JP	MYAPP.PGM1.TITLE.UI	1	\u3053\u3093\u306b\u3061\u306f %s! \u4eca\u65e5 %s \u3000 \u3067 \u3059
17	ja_JP	MYAPP.PGM3.FOOTNOTE.UI	1	\u30e9\u30d0\u30c8\u304c %s \u306b \u4f5c\u6210 \u3055\u308c \u307e \u305f

Figure 4 - Example of a SAS message data set

STEP 3 – RETRIEVING MESSAGES WITHIN SAS CODE

Use the function SASMSG to access and retrieve the message for display within your application. The SASMSG function retrieves a message from the data set created in step 2 based on the current locale and a specified key.

Here is the syntax to retrieve messages within the macro language or within a DATA step or SCL code:

```
%SYSFUNC(SASMSG(BASENAME, KEY, <<QUOTE|Q|DQUOTE|D|NOQUOTE|N> <, substitute1, substitute2, ...>>)
SASMSG("BASENAME", "KEY", <<"QUOTE"|"Q"|"DQUOTE"|"D"|"NOQUOTE"|"N"> <, "subs1", "subs2", ...>>)
```

BASENAME is the name of the data set created from the %SMD2DS macro; it contains the messages.
KEY is the message key. If a key name is specified for a key that does not exist, the key name is returned.

The third parameter of the SASMSG() function allows you to control which type of quotation marks are added to the message or whether quotation marks are added by SASMSG. The default setting is "DQUOTE" if this parameter is omitted. The values for the quoting option are as follows:

N | NOQUOTE : No quotation marks added.
Q | QUOTE : Add single quotation marks around the string.
D | DQUOTE : Add double quotation marks around the string (the default).

The fourth and subsequent arguments to SASMSG are for substitutions into the message text. All substitution parameters specified in SASMSG must be character strings. As many as seven substitutions can be made for one message.

Language of the Messages and Locale Fallback Mechanism

The setting of the LOCALE system option determines the language, and possibly the regional translation, to use for the messages. At run time, the SASMSG function gets the LOCALE value from the LOCALE= system option. A POSIX name in the form ll_rr is returned. The SASMSG function tries to load the key that matches that locale (ll_rr). If no key is found for that locale, the execution falls back to language only (ll). If the code still cannot find a key, SASMSG falls back to the default language. If the message has not been translated for the LOCALE, the default language message text is used. If the key name specified does not exist, the key name is returned.

Example

To demonstrate the SASMSG function, let's run the following code three times with a different setting for the LOCALE option for each execution:

```
%LET MYDS=MYAPP ;
%LET ADATE= %SYSFUNC(today(),nldate.) ;
%LET USER=Steve ;

%PUT %SYSFUNC(SASMSG(&MYDS, myapp.pgml.title.ui, D, &user, &adate)) ;
```

Code box 9. SASMSG function usage examples

When the LOCALE option is set to en_US, this program returns the following English message:

```
Hello Steve! Today's date is February 17, 2012.
```

The same code returns the French translation when the LOCALE option is set to fr_FR:

```
Bonjour Steve! Aujourd'hui nous sommes le 17 février 2012.
```

And it returns a Japanese message when the option is set to ja_JP:

```
こんにちは Steve! 今日は 2012年02月17日 です
```

Macro Language Statements with Special Characters in a Message

The SASMSG function can be used in macro language with the %SYSFUNC macro function. Please note that arguments passed to a function called by the %SYSFUNC macro must not be in quotation marks while arguments passed to the SASMSG function outside of %SYSFUNC must be quoted.

When the SASMSG function is used with the %SYSFUNC macro function, the macro function %NRBQUOTE() is added internally to the returned message. This macro function masks the quotation marks character (') and the double quotation mark character (") when they are not matched:

Ampersand character: &

Messages can contain the ampersand character (&) followed by a valid character other than a blank to reference a macro variable name. The macro variable is resolved with its value, for example:

```
pgm3_osname_ui = Operating System: &osname.
```

At execution time, there are no warnings when the following code is executed:

```
%let osname=%put %sysfunc(sysget(OS) );
%put %sysfunc(sasmsg(&dataset, pgm3_osname_ui));
```

On a Windows NT machine, the generated message is: "Operating System: Windows_NT."

Percent sign character: %

The percent sign character (%) is reserved for format modifiers and could cause unpredictable results if not followed by an appropriate character. The following code is interpreted as format modifiers: %s or %# %1 %2 %3.

To display this character correctly in a message, it must be followed with another % character, for example:

```
msg_note = Increase: %%s%%
```

The execution of the following code produces this message: "Increase: 97%"

```
%put %sysfunc(sasmsg(&dataset,msg_note,d,97)) ;
```

Left and right parenthesis: ()

Because the %NRBQUOTE function is added to the returned message, the parentheses must match. For each open parenthesis, the message must have a close parenthesis.

The SASMSG Function

Like the SASMSG function, the SASMSG function retrieves a message from a data set based on a locale value and a key. However, the SASMSG function does not rely on the current session locale. The locale value must be specified as a parameter.

The syntax to retrieve a message with this function is as follows:

```
%SYSFUNC(SASMSG(BASENAME, KEY, LOCALE, <<Q|D|N> <, substitute1, substitute2, ...>>)
SASMSG("BASENAME", "KEY", LOCALE, <<"Q"|"D"|"N"> <, "subs1", "subs2", ...>>)
```

Example:

```
%LET MYDS=MYAPP ;
%LET ADATE= %SYSFUNC(today(),nldate.) ;
%LET USER=STEVE ;

data _null_ ;
  my_msg = SASMSG("&MYDS", "myapp.pgm1.title.ui", "fr_FR", 'N', "&USER", "&ADATE") ;
  put my_msg= ;
run ;
```

Code box 10. SASMSG function usage example

CONCLUSION

This paper is a brief introduction to the complex field of internationalization within the SAS System. When developing applications for a global market, internationalization is a prerequisite for being successful. It must be part of your initial requirements. The decision regarding which languages to localize for or which markets will be shipped to, is a business decision that will come later in the development schedule, but it will be successful only if the internationalization is done correctly right from the start.

By applying the techniques described in this paper -- choice of appropriate session encoding, use of a locale, use of appropriate string manipulation functions, string externalization -- and with the addition of localized data, the same source code of your SAS program can be fully internationalized and can target a global audience.

REFERENCES

Kiefer, Manfred, and Mebust, Scott. 2007. "Linguistic Collation: Everyone Can Get What They Expect". Technical paper. Available at http://support.sas.com/resources/papers/linguistic_collation.pdf

Kiefer, Manfred. 2010. "Changing language during a SAS session". Technical paper. Available at <http://support.sas.com/resources/papers/LocaleSwitching.pdf>

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

Mebust, Scott, and Bridgers, Michael. 2007. "Creating Order out of Character Chaos: Collation Capabilities of the SAS System". Available at www2.sas.com/proceedings/forum2007/297-2007.pdf

msdn.microsoft.com. "Globalization Step-by-Step: Understanding Internationalization". Available at <http://msdn.microsoft.com/en-us/goglobal/bb688110>

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

ACKNOWLEDGMENTS

We would like to express our gratitude to the people who helped us create this document and provided reviews, especially Manfred Kiefer, Elizabeth Bales, Dan Harrell, and Masahiro Takagi.

CONTACT INFORMATION

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

Mickaël Bouedo
SAS Institute
100 SAS Campus Drive
Cary, NC 27513
E-mail: Mickael.Bouedo@sas.com

Stephen Beatrous
SAS Institute
100 SAS Campus Drive
Cary, NC 27513
E-mail: Steve.Beatrous@sas.com

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.