Multilingual Computing with the 9.1 SAS Unicode Server

Stephen Beatrous, SAS Institute, Cary, NC

ABSTRACT

In today's business world, information comes in many languages and you may have customers and employees in various countries all over the globe. It is very possible that your mission-critical data will be created and stored in more than one language. SAS offers several features that allow you to store and process multilingual data. With SAS 9.1, it is now possible to write a SAS application that processes data from many languages all in the same SAS session. This paper introduces the Unicode support that is provided in SAS 9.1 and discusses several scenarios for how you might use this support to deliver multilingual data to users around the world.

CONCEPTS

You should become familiar with the following basic concepts in order to understand this paper.

- Character Set
- Encoding
- Transcoding
- Unicode
- Legacy Encoding
- SAS DBCS Extensions
- SAS Unicode server

A **character set** is a repertoire of symbols and punctuation marks used in a single language or in a group of languages.

An **encoding** is the association of a unique numeric value with each symbol and punctuation mark in a **character set.** There are two groups or types of encodings: single-byte character set (SBCS) encodings and double-byte character set (DBCS) encodings. SBCS encodings represent each character in a single byte. DBCS encodings require a varying number of bytes to represent each character. A more appropriate term for "DBCS" is multi-byte character set (MBCS). MBCS is sometimes uses as a synonym for DBCS.

SBCS encodings are limited to 256 possible characters. DBCS encodings can represent many more than 256 characters. Beginning with SAS 8.2, each SAS session has one encoding. The encoding for a SAS session is set using the LOCALE or ENCODING option.

Transcoding is the process of converting from one encoding to another.

Unicode is a universal character set that contains the characters in major languages of the world. Other character sets are limited to a subset of the world's languages. Often the subset is regional (for example, Windows Latin 1 (WLATIN1) represents the characters of the US and Western Europe on Windows). UTF8 is one encoding of the Unicode character set in which characters are represented in 1 to 4 bytes.

A **legacy encoding** is one of the DBCS or SBCS encodings which predate the Unicode standards. Legacy encodings are limited to the characters from a single language or a group of languages.

SAS DBCS extensions are an optional supplement to BASE SAS that provide support for DBCS encodings. In SAS 9 the DBCS extensions are available on the SAS software media. When you install SAS, you can choose to install SAS with or without the DBCS extensions.

SAS 9 uses the DBCS extensions to support the UTF8 encoding as a SAS session encoding. In this paper, I will refer to the DBCS system running with a session encoding of UTF8 as **the SAS Unicode server.**

Additional information about these and other SAS international features and options is available in the SAS® 9.1 National Language Support (NLS) Reference book (1) and in the "Base SAS Software" SAS OnlineDoc, Version 9. (2)

INTRODUCTION

BACKGROUND

From Version 5 through Release 8.2, the SAS System was delivered in 2 separate forms: the SBCS system and the DBCS extensions. The SBCS system supports character data in the ASCII and EBCDIC encodings. ASCII and EBCDIC store characters in a single byte. There are multiple extensions to ASCII and multiple versions of EBCDIC, which handle national characters for different regions. For example, the WLATIN1 encoding handles the characters necessary for the languages of Western Europe. The WLATIN2 encoding handles the characters in the languages of Central and Eastern Europe. All ASCII and EBCDIC encodings handle US English characters.

The SAS DBCS extensions support character encodings in which individual characters are represented in multiple bytes. The DBCS system supports SAS customers that use or process data that is stored in languages such as Japanese, Chinese, or Korean.

Both the DBCS and the SBCS SAS systems were designed so that an individual SAS session could represent and process characters within one region or country. That is, an individual SAS session could process only Western European characters, or only Eastern European characters, or only Japanese Characters, or only Chinese characters. In other words, users were unable to process data from all of these languages in a single SAS session.

UNICODE SUPPORT IN SAS 9.1

In 9.1, SAS customers in many regions around the world will use the DBCS extensions in order to support global data (multilingual data which can only be represented in the Unicode character set). With the SAS Unicode server, it is now possible to write a SAS application which processes Japanese data, German data, Polish data, and more, all in the same session. A single server can deliver multilingual data to users around the world.

This paper will discuss the following six scenarios for using the SAS Unicode server.

1. Populating a Unicode database.
2. Using SAS/SHARE® as a Unicode data server.
3. Using thin-client applications with the Unicode data server.
4. Using SAS/IntrNet® as a Unicode compute server.
5. Using AppDev Studio™ as a Unicode compute server.
6. Generating Unicode HTML output using ODS.

The SAS Unicode server is designed to run on ASCII based machines. The SAS Unicode server may be run as a data or compute server or as a batch program.

There are 3 restrictions to the SAS Unicode server.

1. The SAS Display Manger is not supported and if used will not display data correctly.
2. Enterprise Guide® cannot access a SAS Unicode server.
3. You cannot run a SAS Unicode server on MVS (OS/390).

STARTING AND USING A SAS UNICODE SERVER

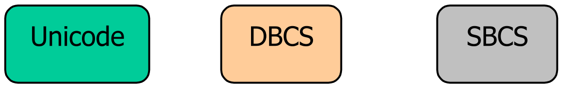
To start a SAS Unicode server you must do two things:

1. Install SAS (release 9.1 or later) with DBCS extensions.
2. Specify ENCODING UTF8 when you start SAS, such as: sas -encoding UTF8

Getting started is that simple. The picture gets complicated when you start thinking about how to convert systems that were written for processing SBCS encoded data, to systems written for processing DBCS Unicode encoded data.

When you use the SAS Unicode server, by default the files that you create and save will store characters in UTF8 encoding. If you read files that were created in other encodings, the data in those files will automatically be converted to UTF8 format using SAS' cross-environment data access (CEDA) feature. (3) In the section of this paper titled "Best Practices" I will discuss how to efficiently use CEDA to bring legacy files into a Unicode format.

The most efficient way to set up a Unicode SAS based application is to have every layer of the application (client, mid-tier, server, and data store) represent strings in Unicode In the diagrams which follow I will use the colors green, tan, and gray to denote Unicode, DBCS, and SBCS, respectively.



ACCESSING AND CREATING DATA

Data can be read into SAS from three external sources.

1. External files
2. SAS Data Libraries
3. DBMS Tables

The SAS Unicode server processes data differently from each source. Tips for processing data from the first two sources are discussed below.

EXTERNAL FILES

External files can be accessed using the FILENAME, ODS, INFILE, or FILE statements.

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 your current SAS session encoding.

When a file contains only character data, use the ENCODING= option on the FILENAME, ODS, INFILE or FILE statement to transcode the data from its original encoding to the current SAS session encoding.
Please see the documentation on these statements for details on the ENCODING= option. (6)

When an external file contains a mix of character and binary data then you must use the KVCT function to convert individual fields from the file encoding to the session encoding.

The KVCT function (2) can be used as shown here:

```
outstring =  kvct(instring,
                  enc_in,
                  enc_out);
```

Where:

instring - input character string.
enc_in - encoding of instring.
enc_out – encoding of outstring.
outstring – results of transcoding instring from enc_in to enc_out.

For example, if you have a WLATIN1 string that you want to convert to UTF8 you could use the following code:

```
out = kvct ( in,
             “WLATIN1”,
             “UTF8”);
```

SAS DATA LIBRARIES

SAS DATA files have an ENCODING attribute in V9. When the file encoding is different from the session encoding, the CEDA facility (3) will automatically transcode character data when it is read and when it is saved.

By default, when you output data from SAS, the new files will be saved using the current session encoding. However, you can also explicitly create a UTF8 data file during an SBCS or DBCS session. The ENCODING=UTF8 option and the OUTENCODING=UTF8 libname option can be used to force SAS 9.1 to create a UTF8 encoded file.

Figure 1 shows how you can use CEDA transcoding to output files to a Unicode data library. This example shows multiple SAS sessions running with the appropriate encoding for a specific region.

Error! Objects cannot be created from editing field codes.

To follow the scenario shown in Figure 1, you must use the ENCODING option on the LIBNAME or dataset specification. The ENCODING option will force the system to transcode character data from session encoding to UTF8 as its being written. (6)

For example, if you are using a French locale, you would do the following:

sas –locale french

```
libname lib 'mult’ outencoding=utf8;
data lib.fra;
   length x $ 20 ;
   x = 'français';
run;
```

If you are using a Japanese locale, you would do the following:

sas -dbcs -dbcslang japanese -dbcstype sjis

```
libname lib 'mult' outencoding=utf8;
data lib.jpn;
   length x $ 20 ;
   x = '•••' ;
run;
```

Both of these code examples enable you to add a Unicode file to the target library.

Figure 2 shows how you can use CEDA to convert SBCS and traditional DBCS files to a UTF8 encoding as the files are read.

Error! Objects cannot be created from editing field codes.

Figures 1 and 2 describe cases where string data is transcoded from a legacy encoding into a UTF8 encoding. This transcoding has one risk. The string data can grow in length when being transcoded from a legacy encoding to a UTF8 encoding. See “Avoiding Character Truncation During Transcoding” in the “Best Practices” section for instructions on reading legacy data or converting legacy data without the risk of truncation.

The scenarios provided in this paper include diagrams that show how to read legacy data into SAS Unicode servers. All of these examples are vulnerable to the risk of string truncation, but you can avoid that risk by properly transcoding your data.

SCENARIO 1: POPULATING A UNICODE DATABASE

The first step in converting an existing database to Unicode or in setting up a new Unicode based system will be to convert all of your data from its legacy encoding to the UTF8 encoding. Once the data is in a Unicode database, there will not be any loss of data when it is read by a Unicode server.

Figure 1 shows how multiple users in your enterprise can simultaneously contribute Unicode data to a central library. Figure 1 presents a distributed model where employees deposit their regional files into a Unicode library.

In some organizations, however, a central database administrator would convert selected data from regional encodings to Unicode. Figure 3 shows how a central administrator could collect data and store it in a Unicode server database.

Error! Objects cannot be created from editing field codes.

To use the model shown in Figure 3, you do not have to use any options if the files being converted are SAS 9 files. If you have files from an earlier release, then you must use a LIBNAME statement or data set option to identify to SAS the current encoding of the input files. The following example demonstrates how you can import Version 8 or Version 9 data.

sas –*encoding UTF8*

```
     /* SAS 9 Data as Input */
data mult ;
   set lat1.data
       lat2.data
       sjis.data ;
run;

  /* SAS 8 Data as Input */
data mult;
   set lat1.data(encoding=wlatin1)
       lat2.data(encoding=wlatin2)
       sjis.data(encoding=sjis) ;
run;
```

SCENARIO 2: USING SAS/SHARE AS A UNICODE DATA SERVER

SAS/SHARE is a product that enables multiple users to access data from a central server. To convert your existing SAS/SHARE server to a SAS Unicode server you must specify the –ENCODING UTF8 config option.

Error! Objects cannot be created from editing field codes.
In Figure 4, clients running SAS with a legacy encoding are able to access the Unicode data from a SAS library or from a DBMS. When the client session uses a legacy encoding (such as Windows Latin1) then there may be some Unicode string data that cannot be represented in the client session. The data will be transcoded from UTF8 encoding to the legacy encoding when it is transferred between the server and the client. If your client is running SAS with a WLATIN1 encoding (to support a language such as French) you will not be able to display a Japanese national character, but you will be able to display any Latin1 based character (French, German, Spanish, etc.).

Those characters which cannot be displayed in the legacy encoding will display as boxes "□" (the standard replacement character). If characters are replaced by the replacement character during transcoding then the data cannot be updated.

If your client is running SAS with a Unicode session encoding you can view all of the data stored on the server.

SCENARIO 3: USING JDBC WITH A UNICODE DATA SERVER

The SAS system is continuously increasing support for industry standard data access protocols such as JDBC. The JDBC interfaces are a data access interface for Java applications. Java supports Unicode string data and therefore, it would be very natural for the SAS Unicode server to function as the data server for Java.

Error! Objects cannot be created from editing field codes.
In SAS 9, many of the new features of the Business Intelligence Platform are written in Java. This includes SAS Management Console and SAS Metadata Server. Note that a SAS Unicode server can be used as a data or a compute server for SAS authored or user authored Java applications.

The SAS ODBC driver and the OLEDB provider currently do not surface Unicode data from a SAS server. This means that thin client applications relying on OLEDB or ODBC for data access will not be able to exploit a SAS Unicode server. We plan to remedy this in a future release.

SCENARIO 4: USING SAS/INTRNET AS A COMPUTE SERVER

The SAS system is often used as a compute server from a non-SAS client. This is another natural fit for the SAS Unicode server.

Error! Objects cannot be created from editing field codes.

The user must specify the –encoding UTF8 config option. There are no changes required to the PROC APPSRV statements (in appstart.sas). There are no changes required for the CGI configuration (in broker.cfg).

When running the app server with a UTF8 encoding, output will be passed to the browser in a UTF8 encoding. The browser will recognize UTF8 data if any of the following are true:

- The browser default encoding is set to Unicode.
- The HTML is preceded by a Unicode byte order mark. This will happen automatically UNLESS the SAS/IntrNet program uses data step put statements to write the HTTP header. Using PUT statements to write the HTTP header has not been recommended for several releases, but many legacy programs still use this old style.
- The HTML contains a <META> tag defining the charset. Any ODS HTML output will contain the <META> tag unless it is explicitly disabled. Other HTML generators (HTML Formatter, put statements, etc.) will not include the <META> tag by default.
- The HTTP header contains a UTF8 charset identifier on the Content-Type record. This can be set in the SAS/IntrNet program with the appsrv_header function.

SCENARIO 5: USING APPDEV STUDIO AS A COMPUTE SERVER

AppDev Studio enables Java programmers to run programs on a SAS server. The programs that run on the server are either SCL programs running with Jconnect or remote objects executed through SAS Integration Technologies.

The Java environment is Unicode enabled. When the object server is a SAS Unicode server and the data sources are Unicode data stores then the AppDev Studio developer can create a truly multilingual application as shown in Figure 7.

ERROR! OBJECTS CANNOT BE CREATED FROM EDITING FIELD CODES.

SCENARIO 6: GENERATING UNICODE HTML OUTPUT USING ODS

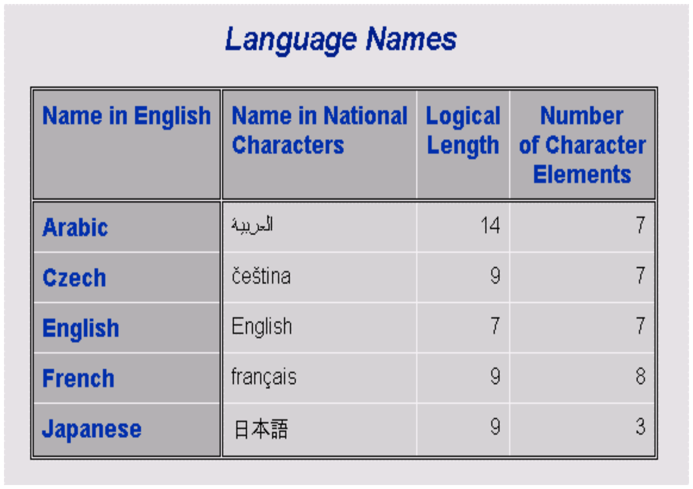
A SAS Unicode server can be used in a batch program to produce ODS output with an encoding of UTF8. At the time of this writing, the following ODS output formats support –encoding UTF8:

- HTML
- XML

Error! Objects cannot be created from editing field codes.

The SAS Unicode server (using a simple PROC PRINT) was used to produce the following report. Note that without the SAS 9.1 Unicode Server it would not have been possible to produce output with this rich set of national characters.

Figure 9: Unicode ODS HTML



Language Names

Name in English	Name in National Characters	Logical Length	Number of Character Elements
Arabic	العربية	14	7
Czech	čeština	9	7
English	English	7	7
French	français	9	8
Japanese	日本語	9	3

BEST PRACTICES AND PITFALLS OF THE SAS UNICODE SERVER

WHAT FORMAT SHOULD I USE FOR MY DATA?

To make the most efficient use of a SAS Unicode compute or data server the data should be stored in Unicode format with an encoding of UTF8. By default, when a file is created it will inherit the current session encoding. Your legacy files will contain character data that is not in Unicode format. One of your first steps in converting an application to run with SAS Unicode server is to convert the data files. As noted above, files can be read by a Unicode Server even if they are not in Unicode format. However, there is a performance cost (as character data is converted when it is read) and there are restrictions (if the file encoding does not match the session encoding the file cannot be updated and cannot utilize index optimization).

You should use the Character Variable Padding engine (CVP) (5) engine described below to convert your files and avoid truncation problems.

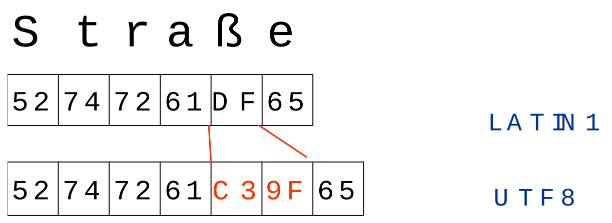
USING THE CVP ENGINE TO AVOID CHARACTER TRUNCATION DURING TRANSCODING

UTF8 encoding requires a varying number of bytes for each character. When you transcode files from a regional encoding to a UTF8 encoding you will likely experience string truncation. You can avoid string truncation problems by padding string data as it is converted from a legacy encoding to a UTF8 encoding.

The following table can help you determine how much expansion to expect.

Bytes in UTF8	Character Sets
1	7bit, US_ASCII Characters
2	Eastern, Central and Western European, Baltic, Greek, Turkish, Cyrillic, Hebrew, and Arabic
3	Japanese, Chinese, Korean, Thai, Indic and certain control characters
4	Some ancient Chinese, special Math symbols (surrogate pairs in UTF16)

For example, assume that you have a 6 byte character field with the value "Straße." In memory the field will look like this:



If CEDA is used to read this field from a Latin1 encoding into a UTF8 encoding then the value will truncate to "Straß" because that is the maximum that can be represented in a 6 byte UTF8 field.

The new CVP engine available in SAS 9.1is a read only engine that will automatically pad character lengths. (5) Using the CVP engine enables you to transcode data to UTF8 without truncation. By default the CVP engine will pad data by a factor of 1.5 when the data is read. For example, a six byte character field becomes a nine byte character field when read by the CVP engine.

The program below will copy all of the input files from X to Y, expand the length of character fields by 1.5 (the default), and transcode the character fields to UTF8 along the way.

```
libname x cvp 'path1';
libname u 'path1' outencoding=utf8;
```

```
proc copy noclone in=x out=u;
   select datasetname;
```

There are 3 things that are particularly important in the previous code example. First, the engine name of CVP should be included on the first LIBNAME statement in order to force strings in the input file to be expanded as they are read.

Second, the OUTENCODING option in the second LIBNAME statement ensures that output files are written in UTF8 encoding. This option is not necessary if the program is being run with a UTF8 session encoding.

Third, by default PROC COPY tries to make an output file with the same attributes as the input file. The NOCLONE option overrides this default.

AVOID TRANSCODING BINARY DATA

Sometimes a data set will contain character fields that are really binary in nature. SAS would corrupt these fields if it transcoded them from the file encoding to the current session encoding. In SAS 9 you can identify binary fields using the TRANSCODE=NO option and prevent truncation problems.

For example, the MXG data set PDB.XTY70D contains many binary fields, e.g. CPUSER0. These fields will be incorrectly transcoded as character data if the file is processed with CEDA. The ATTRIB statement below will preserve the CPUSER0 field while allowing all other character fields to be transcoded.

```
data pdb.xty70d;
   attrib cpuser0 transcode=no;
   set pdb.xty70d;
```

AVOID TRANSCODING ERRORS DURING CEDA

When transcoding data from one encoding to another, an error occurs when the input data contains a character that cannot be represented in the output encoding. Transcoding errors are most common when transcoding from UTF8 to one of the legacy encodings.

If CEDA transcoding errors occur while reading input files, the SAS system will ignore the error as long as the SAS task has no other files open for OUTPUT or UPDATE. Consider the following program:

```
proc print data=cedalib.data;
```

If this program encounters a transcoding error reading CEDALIB.DATA it does no harm. SAS will ignore the error. Now consider this program:

```
data permlib.newdata;
   set cedalib.data;
run;
```

This program will potentially replace a file with bad data. To prevent the risk of data corruption, CEDA treats transcoding errors as an ERROR condition and the data step stops with a NOREPLACE option.

For details on the CEDA rules for processing transcoding errors see "Base SAS Software." SAS OnlineDoc, Version 9. (3)

Transcoding errors can be avoided if all of your data and all of your applications are running Unicode. If you are running a mix of SAS clients in legacy encoding and SAS Unicode servers then you are vulnerable to transcoding errors.

CODING ISSUES: USING THE K STRING FUNCTIONS

If you do not currently use the DBCS SAS system then your SAS programs assume that every character is a single byte in length. You must convert your SAS programs if you want them to support and process UTF8 encoded data. The SAS character functions (for example SUBSTR, INDEX, LENGTH) have DBCS character equivalents (for example KSUBSTR, KINDEX, and KLENGTH). (8)

The following simple example uses two K string functions. This example loops over the characters in a string and assumes that a character can be as much as 4 bytes in length:

```
data _null_ ;
   set merged;
   length ch $ 4 ;
   do i = 1 to klength(maktx) ;
      ch = ksubstr(maktx, i, 1) ;
      put ch=;
   end ;
run;
```

SPDS AND THE SAS UNICODE SERVER

The SAS Performance Data Server® (SPDS) does not support transcoding. This server is built for speed. The SPDS server assumes that the encoding for its client data utilize strings with the same encoding as its server.

The SPDS server can be used as a Unicode data store as long as the files created in the SPDS library were created by SAS Unicode servers and as long as all of the clients expect data in UTF8 encoding.

APPENDIX 1: ENCODING RELATED OPTIONS IN THE SAS SYSTEM

The encoding option is central to understanding how SAS processes string data. The following table summarizes the ENCODING related options in SAS 9. These and other options are discussed in detail in the SAS 9.1 National Language Support (NLS) Reference book. (1)

Encoding Related Options in the SAS system

Option Name	**Purpose**
ENCODING= SAS Configuration option	Specifies the current SAS session encoding
ENCODING= FILENAME option	Specifies the encoding for external files or stream
ENCODING= option in ODS statement	Specifies the encoding for ODS driver. The encoding option is only valid for certain ODS drivers such as HTML, XML, CSV. Some device drivers depend on their own mechanism on support encoding processing.
ENCODING= Dataset option on input / output / update	Specifies the encoding of a SAS Dataset
ENCODING= libname option (INENCODING= for input, OUTENCODEING= for output)	Default encoding for datasets within a library.
CHARSET= option in APPSRV procedure	Establishes the metatag for output streams.
TRANSCODE=YES|NO in ATTRIB statement in DATASTEP (available in 9.1)	Binary character data type. TRANSCODE=NO in ATTRIB statement suppress any transcoding per variable.
TRANSCODE=YES|NO SQL column Modifier	Same as above
CVPMULT= and CVPBYTES= options in CVP Engine	Control the amount of padding.

APPENDIX 2: UNICODE PROCESSING IN THE SAS SYSTEM

There are several Unicode related features of SAS 9. These features are available for SAS sessions running legacy encodings as well as SAS Sessions running with a UTF8 encoding. (1)

- Unicode ENCODING= values for FILENAME and ODS statements. (1)
- Unicode FORMATS and INFORMATS. (7)
- NL formats for displaying currency and date formats matching the user's locale. (7)

CONCLUSIONS

The 9.1 SAS Unicode server introduces a SAS system that can handle data from around the world in a single application. To use the SAS Unicode server you must install SAS (release 9.1 or later) with DBCS extensions and then specify the appropriate encoding when you start SAS. The SAS Unicode server allows you to meet your business need to capture and process national characters from around the world, in one SAS session.

ACKNOWLEDGMENTS

There are many SAS employees from around the world to thank for the Unicode features of SAS. Some of them are:
• • • • • (Shin Kayano)
• • • • (Joji Kobayashi)
Mickaël Bouëdo (Mickael Bouedo)
• • • • • (Atsuko Yoshizawa)
Paula Smith (Paula Smith)
Manfred Kiefer (Manfred Kiefer)
Jack Wallace (Jack Wallace)

REFERENCES

1. SAS(R) 9.1 National Language Support (NLS) Reference. SAS Institute Inc., Cary, NC. SAS.
2. "Base SAS Software." SAS OnlineDoc, Version 9.1 2003 CD-ROM. SAS Institute Inc., Cary, NC. SAS.
3. Cross-Environment Data Access (CEDA). "Base SAS Software." SAS OnlineDoc, Version 9. 2003. CD-ROM. SAS Institute Inc., Cary, NC. SAS.
4. Cross-Environment Data Access (CEDA). SAS Institute Inc., Cary, NC.SAS Available at: http://support.sas.com/rnd/migration/planning/files/ceda.html.
5. Character Variable Padding (CVP). "Base SAS Software." SAS OnlineDoc, Version 9.1 2003. CD-ROM. SAS Institute Inc., Cary, NC. SAS.
6. Encoding. "National Language Support (NLS) Reference." SAS OnlineDoc, Version 9.1 2003. CD-ROM. SAS Institute Inc., Cary, NC. SAS.
7. NLS Formats. "National Language Support (NLS) Reference." SAS OnlineDoc, Version 9.1 2003. CD-ROM. SAS Institute Inc., Cary, NC. SAS.
8. NLS Functions. "National Language Support (NLS) Reference." SAS OnlineDoc, Version 9.1 2003. CD-ROM. SAS Institute Inc., Cary, NC. SAS.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at: steve.beatrous@sas.com