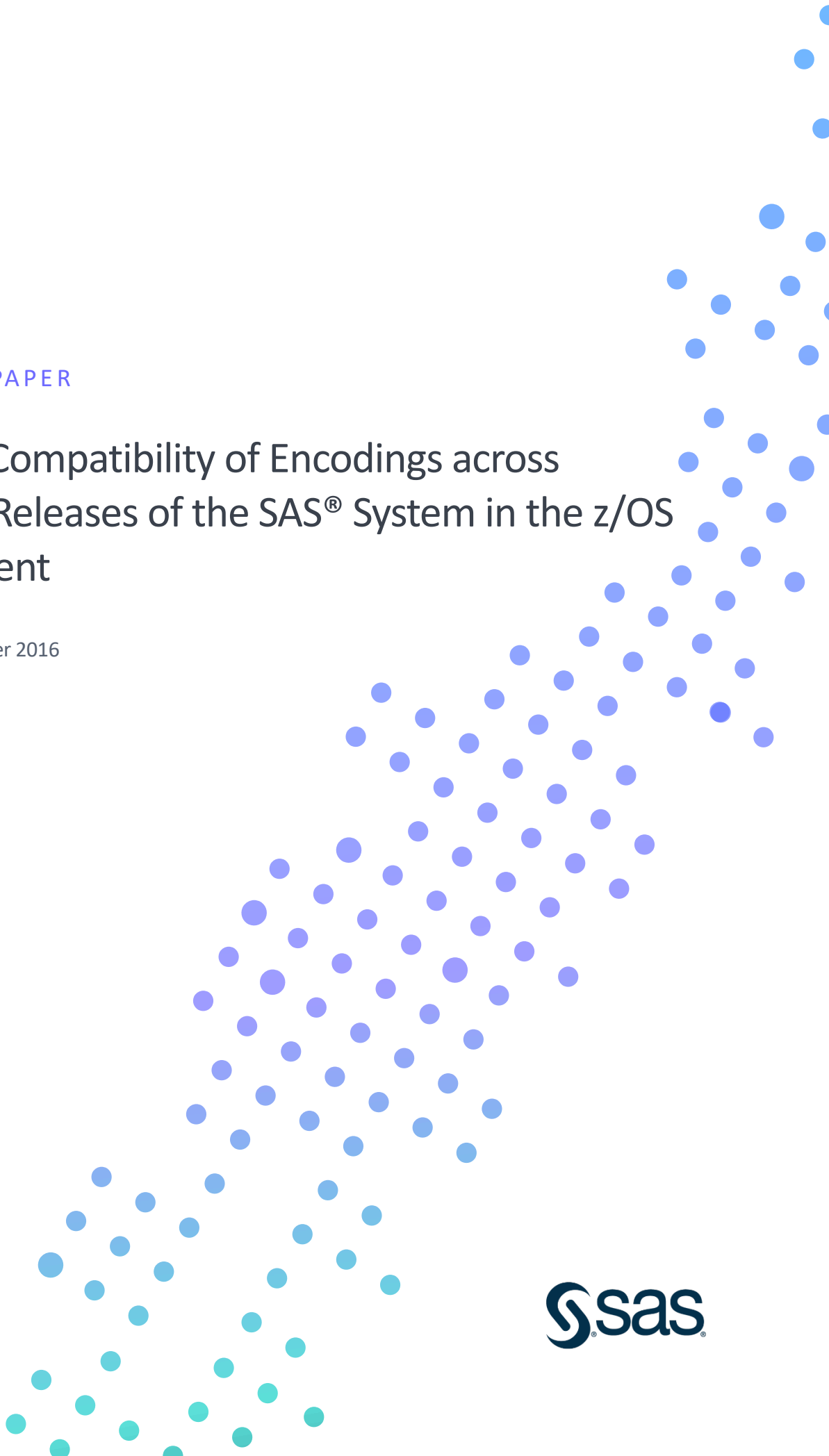
TECHNICAL PAPER

Ensuring Compatibility of Encodings across Different Releases of the SAS® System in the z/OS Environment

Last update: October 2016



Contents

Introduction

Beginning with SAS® 8.2, Base SAS® software has provided separately encoded versions for various locales1 . It has also provided the NLSCOMPATMODE option to deal with legacy code.

The default for the NLSCOMPATMODE option is NONLSCOMPATMODE. The NONLSCOMPATMODE option specifies that data is processed in the session encoding, including reading and writing external files as well as processing SAS syntax and user data. The session encoding is the encoding set in the ENCODING= system option.

The NLS Compatibility mode specified with the NLSCOMPATMODE option will be deprecated because continued use of NLSCOMPATMODE perpetuates uncertainties about character handling. In particular, users writing SAS code to run in NLSCOMPATMODE must remember to use replacement characters rather than the characters they see in the SAS documentation. Also, the following warning is displayed when the SAS system is started in such mode:

WARNING: SAS has been started in NLS compatibility mode with the NLSCOMPATMODE option. This option will no longer be supported after this release. For more information, contact a SAS representative or Technical Support.

This document describes how different encodings affect SAS applications at different release levels. It also explains how migrating your SAS applications so that they use NONLSCOMPATMODE can keep these applications from being affected in the future

The Problem with EBCDIC

Extended Binary Coded Decimal Interchange Code (EBCDIC) is a family of related encodings used by IBM. Unlike most 8-bit encodings, it is not compatible with ASCII. Rather, characters are coded according to the historical needs of punch-card machines. Each EBCDIC variation, known as a code page, is identified by a Coded Character Set Identifier, or CCSID. The characters in the basic set (a-z, A-Z, 0-9, and so on) are mapped to the same code points on all EBCDIC code pages (these are called invariant characters). The rest of the code points can be used for different national and special characters (these are called variant characters), dependent upon for which country and language a code page was developed.

What are Variant Characters?

The following characters are considered variant because they can have different code positions in various EBCDIC variations:

! # $ @ \ [ ] ^ `{ } | ~

Note: New terms, noted on first appearance in this paper, are explained in the Glossary.

These characters exist in every encoding, but their hex values might change from one encoding to another, as shown in the following table:

Table 1. *Variant EBCDIC characters*

Character	1047	838	870	1025	1141	1142	1143	1144	1145	1146	1147	1148
!	5A	5A	4F	4F	4F	4F	4F	4F	BB	5A	4F	4F
#	7B	7B	7B	7B	7B	4A	63	B1	69	7B	B1	7B
$	5B	5B	5B	5B	5B	67	67	5B	5B	4A	5B	5B
@	7C	7C	7C	7C	B5	80	EC	B5	7C	7C	44	7C
\	E0	E0	E0	E0	EC	E0	71	48	E0	E0	48	E0
[	AD	49	4A	4A	63	9E	B5	90	4A	B1	90	4A
]	BD	59	5A	5A	FC	9F	9F	51	5A	BB	B5	5A
^	5F	69	5F	5F	5F	5F	5F	5F	BA	BA	5F	5F
`	79	79	79	79	79	79	51	DD	79	79	A0	79
{	C0	C0	C0	C0	43	9C	43	44	C0	C0	51	C0
}	D0	D0	D0	D0	DC	47	47	54	D0	D0	54	D0
|	4F	4F	6A	6A	BB	BB	BB	BB	4F	4F	BB	BB
~	A1	A1	A1	A1	DC	DC	DC	58	BD	BC	BD	A1

The problem of EBCDIC encoding for SAS is that variant characters are part of the SAS language syntax itself. For example, the dollar sign ($) is used for character formats, and the curly braces ({}) or the square brackets ([]) are used for array statements.

In addition, the newline (\n) character is treated as a variant character, even though it is at the same code point in all EBCDIC encodings. This treatment results from the possible variations in transcoding from the newline character to the line-feed character. For more details, see the section, What is the Difference between Open Edition EBCDIC and Traditional EBCDIC?

What is Compiler Encoding?

SAS uses compiler encoding, which is the encoding that is used to compile the SAS System for processing external data and SAS syntax. This compiler encoding is the same encoding that is used for U.S. English (code page 1047), and it is used if you do not change the locale or encoding of the SAS session. Before SAS 8.2, Base SAS software used compiler encoding for all data processing.

Keeping Compatibility with the Past

Beginning with SAS® 8.2, Base SAS® software has not only provided separately encoded versions for various locales but it also provided the NLSCOMPATMODE option to deal with legacy code. In NLSCOMPATMODE mainframe programmers could use replacement characters for variant characters that are part of the SAS language syntax itself. For example, programmers in the United Kingdom have used £ as a replacement for $. This worked fine in programs such as the following:

```
data mylib.employee;
         input name £char8. +2 income comma6.;
               datalines;
Peterson 21,000
Morgan 17,132
 ;
run;
```

It worked because compiler encoding was used for all data processing and the hex value for the characters $ and £ are identical (0x5b), whereas the display depends on the terminal emulator settings.

Consequently, z/OS media was made available in the following encoded versions, which support multiple locales and regions. (For “Languages and Installation Codes”, see the Configuration Guide for SAS® 9.4 Foundation for z/OS® at http://support.sas.com/documentation/installcenter/en/ikfdtnmvscg/66194/PDF/default/config.pdf).

SAS System installation media VOLSERs and certain installed SAS System filenames (such as SASHELP) contain a two-character code that identifies the encoding and locales that are supported—for example, W3 for Austria and Germany (CP1141) or WB for International (CP1148). See the following table and “Mapping of Encodings by Country” in Chapter 2 of the Configuration Guide for SAS® 9.4 Foundation for z/OS® for further details at http://support.sas.com/documentation/installcenter/en/ikfdtnmvscg/66194/PDF/default/config.pdf. User data files should match the encodings below.

Table 2: *Encodings and installation code*

Encode value	Country	Main encoding	Default locale	Alternate locales
C0	Poland	OPEN_ED-870	Polish_Poland	Albanian_Albania, Bosnian_ BosniaHerzegovina, Croatian_Croatia, Czech_CzechRepublic, Hungarian_Hungary, Romanian_Romania, Slovak_Slovakia, Slovenian_Slovenia
F0	Thailand	OPEN_ED-838	Thai_Thailand	
R0[2]	Russia	OPEN_ED-1025	Russian_Russia	Bulgarian_Bulgaria, Serbian_Serbia, Russian_Ukraine, Macedonian_Macedonia
	Greece	OPEN_ED_875		
W0	United States	OPEN_ED-1047	English_UnitedStates	Afrikaans_SouthAfrica English_Australia English_Botswana English_Canada English_Caribbean English_Jamaica English_NewZealand English_Philippines English_SouthAfrica Dutch_Netherlands French_Canada Icelandic_Iceland Indonesian_Indonesia Malay_Malaysia Portuguese_Brazil Portuguese_Portugal Spanish_Argentina Spanish_Bolivia Spanish_Chile Spanish_Colombia Spanish_Ecuador Spanish_ElSalvador

Encode value	Country	Main encoding	Default locale	Alternate locales
				Spanish_Mexico Spanish_Nicaragua Spanish_Panama Spanish_Paraguay Spanish_Peru Spanish_PuertoRico Spanish_UnitedStates Spanish_Uruguay Spanish_Venezuela
W3	Germany	OPEN_ED-1141	German_Germany	German_Austria
W5	Denmark	OPEN_ED-1142	Danish_Denmark	Norwegian_Norway
W6	Finland	OPEN_ED-1143	Finnish_Finland	Swedish_Sweden
	Estonia	OPEN_ED-1122		
W7	Italy	OPEN_ED-1144	Italian_Italy	
W8	Spain	OPEN_ED-1145	Spanish_Spain	
W9	United Kingdom	OPEN_ED-1146	English_ UnitedKingdom	English_ Hongkong English_India English_Ireland English_Singapore
WA	France	OPEN_ED-1147	French_France	French_Luxembourg
WB	Belgium	OPEN_ED-1148	French_Belgium	Dutch_Belgium French_Switzerland German_Switzerland Italian_Switzerland
	Vietnam	OPEN_ED-1130		
WU	Japan	OPEN_ED-1047	English upcase-only build for Japanese customers	

The NONLSCOMPATMODE option turns off NLS compatibility mode. When NONLSCOMPATMODE is set, all character data (including external data) is processed by using the encoding that is set in the ENCODING= option. In this mode, the encoding that SAS uses to process character data is the encoding that is specified explicitly by the ENCODING= system option or that is set implicitly by the LOCALE= system option. This behavior is the default behavior in SAS®9 3 .

Issues Related to Migrating Legacy Code

Let's have another look at our example from above. Generations of mainframe programmers in the United Kingdom have used £ as a replacement for $:

```
data mylib.employee;
     input name £char8.   +2 income comma6.;
     datalines;
Peterson 21,000
Morgan 17,132
;
run;
```

It worked because compiler encoding was used for all data processing and the hex value for the characters $ and £ are identical (0x5b), whereas the display depends on the terminal emulator settings.

As mentioned, continued use of NLSCOMPATMODE perpetuates uncertainties about character handling. In particular, it requires users to know which replacement characters to use for the encoded image that is available to them instead of being able to simply edit the code. Replacement characters use the same code values as the ones in compiler encoding but display a different glyph. For example, a Danish user has to use the characters "æ" and "å" for the curly braces "{" and "}" in an array statement, a Spanish user a "Ñ" for the hash sign "#", a German user a "Ü" for the exclamation mark "!" and so on.

This can be illustrated for some selected code pages in the table below:

Table 3: *Code Point Discrepancies among EBCDIC Encodings*

EBCDIC Code Point	OPEN_ED-1047	OPEN_ED-1141	OPEN_ED-1142	OPEN_ED-1143	OPEN_ED-1145	OPEN_ED-1146	OPEN_ED-1148
5A	!	Ü	¤	¤	]	!	]
7B	#	#	Æ	Ä	Ñ	#	#
5B	$	$	Å	Å	$	£	$
7C	@	§	Ø	Ö	@	@	@
E0	\	Ö	\	É	\	\	\
AD	[4	Ý	Ý	Ý	Ý	Ý	Ý
BD	]	¨	¨	¨	¨	¨	¨
5F	^	^	^	^	¬	¬	^
79	`	`	`	é	`	`	`
C0	{	ä	æ	ä	{	{	{
D0	}	ü	å	å	}	}	}
4F	|	!	!	!	|	|	!
A1	~	ß	ü	ü	¨	~	~

Continued use of NLSCOMPATMODE also contradicts the goal of effectively exchanging data with other software. In addition, you must use NONLSCOMPATMODE if you run SAS Business Intelligence solutions and any other new products and features under z/OS.

The NONLSCOMPATMODE option does not use compiler encoding, in which £ is recognized as a character format. This means that legacy code (such as the code in the preceding example) will not run any more. To enable the code to run, you have to replace the £ character (0x5b) with a $ character from session encoding (0x4a in this case, from code page 1146) because $ is the only recognized format identifier (see the section Migration Methods: Aids and Tools).

Alternatively, a new set of translation tables was added that override the way SAS® interprets characters used in syntax. These translation tables map variant characters to session encoding so that you can run your legacy programs using the default NONLSCOMPATMODE setting.

The translation tables are named using the 4-byte version of the SAS® encoding name plus "_SNP" on the end. To set up your SAS® environment to map compiler characters to session encoding, assign the name of the translation table to the 6th slot in the TRANTAB option.

For example, the encoding for English (United Kingdom) is open_ed-1146 and the 4-byte name is eo46. So, the scanner table would be set to eo46_snp. You can specify the TRANTAB= option for English (United Kingdom) like this in a configuration file or at SAS invocation:

```
TRANTAB=(eo46wlt1,wlt1eo46,eo46_ucs,eo46_lcs,eo46_ccl, eo46_snp,,,eo46_scc).
```

For information about the 4-byte SAS encoding names, see "SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" in the SAS® 9.4 National Language Support (NLS): Reference Guide, Third Edition. Please note that the 4-byte names for the open edition encodings begin with 'eo'.

Please note, that this special trantab does not transcode strings between quotes, for instance in a TITLE statement.

Hence in this example:

TITLE 'PRIMERA ÑBYVAR1 SEGUNDA ÑBYVAR2';

the translation table does not transcode the "Ñ" to a hash sign "#". This has to be done manually or programmatically.

There is one important exception: **variant characters** that appear in **physical operating-system** data set **names** (such as data set names, volume serial number names, or System Managed Storage data, storage, and management class names) must **always** be specified with the **code points** used internally **within the operating system**. In this respect, SAS behaves in a manner identical to the native interfaces (JCL, ISPF, Unix shell, etc.) provided with z/OS. Whatever code points these interfaces display for an OS resource name, the same code points should be specified within SAS. The interfaces perform no transcoding. For example, for the 8th character in the data set name userid.$.myfile, a British customer will specify the character £ (which is at code point 0x5B in the English-UK 1146 code page) to reference the data set userid.$.myfile. See Appendix 3 "Encoding for z/OS Resource Names" in the SAS® 9.4 Companion for z/OS at
http://support.sas.com/documentation/cdl/en/hosto390/68955/PDF/default/hosto390.pdf

This behavior was introduced for SAS 9.3. A user can revert to the SAS 9.2 method of handling OS resource names by setting the TKMVSENV environment variable TKOPT_ENV_ENCODING_PATH to a value of "open_ed-1047".

What is the Difference between Open Edition EBCDIC and Traditional EBCDIC?

The characters that are used to explicitly indicate line boundaries are represented differently on different

platforms, but they also show ambiguous behavior even on the same platform.

Software that runs under ASCII operating environments requires the end of the line to be specified by the line-feed character. When data is transferred from z/OS to a machine that supports ASCII encodings, formatting problems can occur, particularly in HTML output, because the EBCDIC newline character is not recognized. SAS supports two sets of EBCDIC-based encodings for z/OS.

The encodings that have EBCDIC in their names (for example, EBCDIC1141 or EBDCIC1144) use the traditional mapping of the EBCDIC line-feed character to the ASCII line-feed character. This mapping can cause data to appear as one stream. The ENCODING= option is set to the traditional EBCDIC encoding for the locale for NLS compatibility mode (NLSCOMPATMODE).

The Open Edition EBCDIC encodings have OPEN_ED in their names (for example, OPEN_ED-1141 or OPEN_ED1144). These encodings use the line-feed character as the end-of-line character. When the data is transferred to an operating environment that uses ASCII, the EBCDIC newline character maps to an ASCII line-feed character. This mapping enables ASCII applications to interpret the end-of-line correctly, resulting in better formatting. The ENCODING= option is set to an Open Edition version of the EBCDIC encoding when you are running SAS with NLS compatibility mode turned off (NONLSCOMPATMODE)

Hints and Tips

- SCL code that is compiled by using the NLSCOMPATMODE option still runs when the NONLSCOMPATMODE option is used. If you need to make changes, however, you must transcode the source code and recompile it using the NONLSCOMPATMODE option.

- Customized menus associated with an FSEDIT window must be recompiled with variant characters in a session encoding and when the NONLSCOMPATMODE option is used.

- Beginning with the third maintenance release for SAS 9.2, the following warning message appears in the log to warn z/OS customers:

```
WARNING: SAS has been started in NLS compatibility mode with the
NLSCOMPATMODE option. This option will be deprecated in a future
release of SAS and NLS compatibility mode no longer be supported. For
more information, contact a SAS representative or Technical Support.
```

This means that you will not receive any hot fixes or future fixes, and features that were added in SAS®9 are not guaranteed to work in NLSCOMPATMODE. You must always check to see if your code still runs in a new release. Therefore, you should migrate your legacy code per the information in the following section, Migration Methods: Aids and Tools.10

- MXG code5 that is heavily used with the SAS® IT Resource Management solution is only delivered in compiler encoding (in fact, in code page 1047). You can use the SASAUTOS= option to enable concatenated autocall libraries with different encodings. This also involves specifying FILENAME statements to define the encodings, as shown in this example:

```
filename sourcelib 'sasprod.mxg.autocall.cass'
                  encoding='open_ed-1047';
```

```
filename concat 'sastjv.autocall.sas'
                encoding'open_ed-1143';
options sasautos=(concat, sourclib, sasautos);
```

Migration Methods: Aids and Tools

Catalog and Data Sets

The easiest way to migrate a catalog or a data set is to use the MIGRATE procedure. When remote library services are not used, migrated data files take on the data representation and encoding attributes of the library, and data sets take on the attributes of the host on which the target library is located. This means that legacy data that do not have an encoding attribute will get one. For more details about using PROC MIGRATE, see the documentation that is available about migration on the "Migration: Data" Web page on the SAS Customer Support Site (support.sas.com/rnd/migration/planning/files/).

You can also use the CPORT and the CIMPORT procedures to migrate your data. You use PROC CPORT to create a transport file, and then you use PROC CIMPORT to import the catalog or data set. Imported data files take on the data representation and encoding attributes of the SAS session they are imported to. For external files you can use the same method, or you can use the ENCODING= option in the FILENAME statement.

Unfortunately, this method does not allow doing context-sensitive transcoding. So if you have mixed data and code, you might have to change national characters in textual data manually or programmatically (see the following section).

Itemstore Files (Registry and Template)

Use the source code to build the itemstore files in the new encoding. If you do not have the source, you can dump the registry into a flat file and rebuild it with the EXPORT= option in the REGISTRY procedure, as follows:

```
filename regfile "file-name";
proc registry listreg=libname.registryname export=regfile;
run;
```

Then, re-import the file using the IMPORT= option:

```
proc registry import="file-name";
run;
```

PROC TEMPLATE has similar features to assist with this issue. In either case, make sure the session encoding is set correctly or specify the encoding in the FILENAME statement so that your data is transcoded properly when it is imported.
For additional details, see "The REGISTRY Procedure" and "The TEMPLATE Procedure" in Base SAS® 9.4 Procedures Guide at http://support.sas.com/documentation/cdl/en/proc/68954/PDF/default/proc.pdf.

Raw Source Code

Imagine you have stored source code to create an employee table with Swedish names in a partitioned data set (PDS) member called mydata.source(employee):

```
data employee;
         input name Åchar9. Ö10 income commax6.;
             datalines;
Bergman 21.000
Holmberg 17.132
Lindgren 22.500
Sandström18.000
Åkerlund 22.000
Öst 19.800
;
run;
```

Now you are running a SAS session with the LOCALE= option set to Swedish. This means your session encoding is EBCDIC1143.

After you specify the command **`INC 'MYDATA.SOURCE(EMPLOYEE)' ENCODING='OPEN_ED-1047'`**, your code is transcoded from EBCDIC1047 to EBCDIC1143. Then, the code will appear as follows:

```
data employee;
         input name Åchar9. Ö10 income commax6.;
             datalines;
Bergman 21.000
Holmberg 17.132
Lindgren 22.500
Sandström18.000
Åkerlund 22.000
Öst 19.800
;
run;
```

Here you read in code with the ENCODING= option to convert it to session encoding. Of course, this method cannot perform context-sensitive transcoding. This means that if you have code with embedded text and comments (such as the sequential input in a DATALINES statement, as shown in the example below), all characters are transcoded from one EBCDIC code page to another. (Strings that are passed to the operating system are another example of such text.) As a result, filenames, paths, user names, and passwords might have characters or syntax modified when they should not be modified.

You have to change the Swedish characters either manually or programmatically. SAS Technical Support can provide you with a REXX procedure to make the change programmatically. For more information, contact SAS Technical Support.

Conclusion

In order to achieve a consistent behavior of applications across platforms, you should be aware of the following:

- A SAS®9 release should be used with an encoded edition that matches your operating system's encoding.
- Under z/OS, SAS®9 uses the NONLSCOMPATMODE option by default.
- The NLSCOMPATMODE option has become obsolete because continued use of the option perpetuates uncertainties about character handling, and it contradicts the goal of effectively exchanging data with other software.
- You need to migrate legacy code by using one of the previously described migration methods

References

Hart, Edwin, ed. 1989. ASCII and EBCDIC: Character Set and Code Issues in Systems Application Architecture. Chicago, Illinois: SHARE Inc., SSD #366.

IBM Corporation. 2004. Personal Communications for Windows Host Code Page Reference. 1 st ed. Program number: 5639–I70. Armonk, NY: IBM Corporation. Available at publib.boulder.ibm.com/infocenter/pcomhelp/v5r9/topic/com.ibm.pcomm.doc/reference/pdf/hcp_referenceV58.pdf.

SAS Institute Inc. 2001. TS-653 - Globalization and National Language Support for Your Version 8.2 SAS Environment. Cary, NC: SAS Institute Inc. Available at support.sas.com/techsup/technote/ts653.pdf.

SAS Institute Inc. 2016. Configuration Guide for SAS® 9.4 Foundation for z/OS®, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/installcenter/en/ikfdtnmvscg/66194/PDF/default/config.pdf.

SAS Institute Inc. 2015. SAS® 9.4 National Language Support (NLS): Reference Guide. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/nlsref/67964/PDF/default/nlsref.pdf.

SAS Institute Inc. 2015. SAS® 9.4 Companion for z/OS, Fifth Edition. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/hosto390/68955/PDF/default/hosto390.pdf.

Glossary

Character set

The set of characters and symbols that are used by a language or group of languages. A character set includes national-language characters (characters that are specific to a language as it is written in a particular nation or group of nations), special characters (such as punctuation marks), the unaccented Latin characters A-Z, the digits 0-

9, and control characters that are needed by the computer.

code page

The representation of a character set that associates a hexadecimal value with each character. The term code page originated from IBM's EBCDIC-based mainframe systems, but many vendors use this term including Microsoft, SAP, and Oracle Corporation

Encoding

A set of 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. An encoding results from applying an encoding method to a specific character set. Groups of encodings that apply the same encoding method to different character sets are sometimes referred to as families of encodings. For example, German EBCDIC is an encoding in the EBCDIC family, Windows Cyrillic is an encoding in the Windows family, and Latin-1 is an encoding in the ISO 8859 family. There are two 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 used as a synonym for DBCS.

Glyph

A glyph is the most basic element of a typeface or font that carries meaning in the text of a writing system. For example, the Z character can be represented by a number of different glyphs--boldface, italic, or in varying font styles, all of which represent the letter "Z."

Legacy encoding

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

Locale

A value that reflects the language, local conventions, and culture for a geographic region. Local conventions can include specific formatting rules for dates, times, and numbers, and a currency symbol for the country or region. Collating sequences, paper sizes, and conventions for postal addresses, and telephone numbers are also typically specified for each locale. Some examples of locale values are French_Canada, Portuguese_Brazil, and English_USA.

Session encoding

The encoding that is used for your SAS session. It is specified explicitly by the ENCODING= system option or set implicitly by the LOCALE= system option.

Transcoding

The process of converting the contents of a SAS file from one encoding to another encoding. Transcoding is necessary if the session encoding and the file encoding are different, such as when transferring data from a Latin-1 encoding in UNIX environments to a German EBCDIC encoding on an IBM mainframe.

Translation table

A SAS catalog entry that is used to map data from one encoding to another encoding. Translation tables or trantabs are specific to the operating environment.

Release Information Content Version: 1.0 October 2016

To contact your local SAS office, please visit: sas.com/offices