Technical Paper

Processing Multilingual Data with the SAS® 9.2 Unicode Server

Content provider for Processing Multilingual Data with the SAS 9.2 Unicode Server was Manfred Kiefer, Globalization Specialist, SAS Heidelberg.

Table of Contents

Abstract

In a global economy, data often comes from various countries using different languages and scripts.

This data can be stored in a relational database management system (for example: Oracle, DB2, Informix, and MS-SQL), in Excel spreadsheets, or in plain text files. Also, the data can come from different platforms or operating systems, such as Windows, AIX, HP-UX, Linux, Solaris, or other UNIX derivatives.

Instead of keeping the data in separate databases, it is more efficient to maintain a central repository that can support a global set of languages. With the SAS® *Unicode*[1] server, which is a SAS session *encoding* of *UTF-8*, it is possible to store, process, and deliver multilingual data.

An Introductory Example

The following example is a simple sample scenario.

A company has received name and address source data from several Western European countries and imported it to their database. The data is generated into enhanced contact details for direct mail communications in SAS.

The company intended to process names and addresses from Eastern European and Asian countries. However, some or all of the characters have been corrupted after importing to SAS because they did not use an appropriate encoding for the particular languages. They could change the encoding each time they process a language that belongs to a different group, by running SAS with ENCODING=WLATIN1 for French data, then rerun the script with ENCODING=WLATIN2 for Polish, and ENCODING=SHIFT-JIS for Japanese. The result is multiple output data sets. However, a better approach is to use an encoding that deals with all language groups (for example: Western, and Eastern European and Japanese) together and generate a single output data set. This encoding is UTF-8, a form of Unicode, a universal encoding that can handle characters from all possible languages.

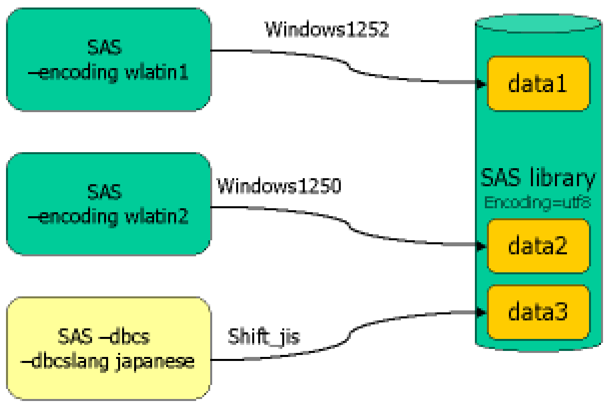


Figure 1: Transcoding during OUTPUT

Before we go into more detail, let us have a closer look at the installation and configuration of the SAS Unicode server.

Installing and Configuring a Unicode Server

SAS® 9.2 expands the scope and capabilities of National Language Support (NLS). Many features are easier to use and more intuitive.

The SAS Deployment Manager does not prompt the user for locale; the default language and locale is determined automatically by SAS Deployment Manager based on the user's regional settings or the operating system's locale. This language and locale is associated with the .sas filetype on Windows (image invoked when a .sas file is double-clicked).
Only the appropriate language(s) for the locale and encoding settings is installed. If the locale is an English locale, all languages that the user chooses are installed. Configuration errors occur if there are illegitimate combinations. English is the default in the situation where the language for the default locale for MultiVendor Architecture is not installed.

By default, SAS Deployment Manager always installs the English product and DBCS and Unicode support with the English product. English, DBCS, and UTF-8 are listed on the language selection menu, but are checked and grayed out.

On Windows, shortcuts for all installed languages are provided on the SAS programs menu. Example:

Programs ► SAS ► SAS 9.2 (English)
- **► SAS 9.2 (default language)**
- **► SAS 9.2 (Additional Languages)**
  - **► SAS 9.2 (English (DBCS))**
  - **► SAS 9.2 (additional language #1)**
  - **► SAS 9.2 (additional language #2)**
  - **... ....**
- **► SAS 9.2 License Renewal & Utilities**
  - **► Renew SAS Software**
  - **► SAS 9.2 (Unicode Server)**

On UNIX, startup scripts/config files are created for all installed languages. The default "sas" startup script has a link to the default language. In order to change the default language, the user simply changes this link to point to the appropriate language script. This is what 9.1.3 customers should already be accustomed to doing. In the above case the "sas" script would have a link to "bin/sas_fr". SAS Foundation is the only SAS product which requires an invocation point for each language. To start a SAS Unicode session, you just need to use the script sas_u8 that is located in the !SASROOT/bin directory.

In SAS 9.2, a set of TrueType fonts are installed and registered automatically with the SAS system. This feature provides an internationalized set of TrueType fonts on every operating system where SAS can be installed.

Table 1: Fonts Licensed from Monotype Corp.

Font Name	Font Filename	Size
Albany AMT (regular)	`saswalr.ttf`	166K
(bold)	`saswalb.ttf`	168K
(italic)	`saswali.ttf`	177K
(bold italic)	`saswalbi.ttf`	171K
Cumberland AMT (regular)	`saswcur.ttf`	122K
(bold)	`saswcub.ttf`	119K
(italic)	`saswcui.ttf`	145K
(bold italic)	`saswcubi.ttf`	133K
Thorndale AMT (regular)	`saswthr.ttf`	207K
(bold)	`saswthb.ttf`	188K
(italic)	`saswthi.ttf`	194K
(bold italic)	`saswthbi.ttf`	183K
Monotype Sorts	`sasgsort.ttf`	81K
Symbol MT	`sasgsymb.ttf`	122K
Monotype Sans WT SC	`sasssans.ttf`	34636K
Monotype Sans WT TC	`sastsans.ttf`	29556K
Monotype Sans WT J	`sasjsans.ttf`	27262K
Monotype Sans WT K	`sasksans.ttf`	28173K
Thorndale Duospace WT SC	`sassserf.ttf`	41043K
Thorndale Duospace WT TC	`sastserf.ttf`	35571K
Thorndale Duospace WT J	`sasiserf.ttf`	35786K
Thorndale Duospace WT K	`saskserf.ttf`	34802K

Eight additional fonts were licensed from Ascender Corp. in order to support Asian languages in their native encodings:

Table 2: DBCS Fonts Licensed from Ascender Corp.

Font Name	**Font Filename**	**Size**
Japanese o MS PGothic o MS PMincho	`Msgothic.ttc` `Msmincho.ttc`	9086636K 9141172K
Simplified Chinese o Sim Hei o Sim Sun	`Simhei.ttf` `Simsun.ttc`	10046900K 10506316K
Traditional Chinese o HeiT o PMingLiU	`Heit.ttf` `Mingliu.ttc`	21164128K 10594028K
Korean o Gulim o Batang	`Gulim.ttc` `Batang.ttc`	13528836KK 16268716K

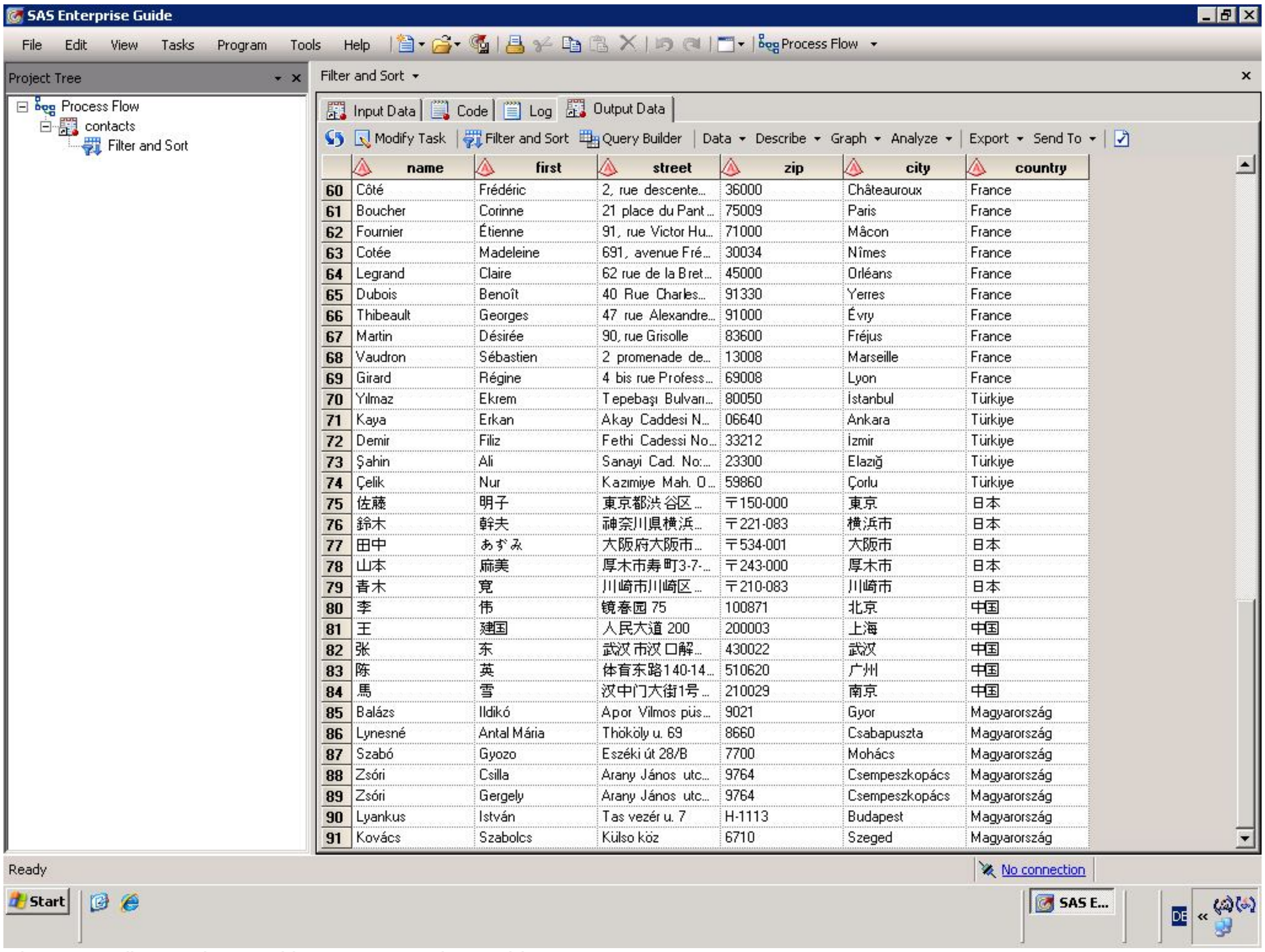
ODS style templates read font names from the SAS registry. There are predefined settings for Japanese, Chinese (both Traditional and Simplified), Korean and Thai. For languages using an SBCS encoding, "Thorndale AMT" is used. Customers can change fonts to be used with ODS by updating these keys.

Viewing Data

The SAS windowing environment[2] on UNIX does not support Unicode, it does not display data correctly. The SAS windowing environment on Windows has limited support for Unicode, therefore you can enter data using the Enhanced Editor or the Program Editor. You will see the following warning message in the log:

```
WARNING: Display of UTF-8 encoded data is not fully supported by the SAS
Display Manager System.
```

We recommend using SAS® Enterprise Guide® as an easy to use front end to the SAS Unicode server.



	name	first	street	zip	city	country
60	Côté	Frédéric	2, rue descente...	36000	Châteauroux	France
61	Boucher	Corinne	21 place du Pant...	75009	Paris	France
62	Fournier	Étienne	91, rue Victor Hu...	71000	Mâcon	France
63	Cotée	Madeleine	691, avenue Fré...	30034	Nîmes	France
64	Legrand	Claire	62 rue de la Bret...	45000	Orléans	France
65	Dubois	Benoît	40 Rue Charles...	91330	Yerres	France
66	Thibeault	Georges	47 rue Alexandre...	91000	Évry	France
67	Martin	Désirée	90, rue Grisolle	83600	Fréjus	France
68	Vaudron	Sébastien	2 promenade de...	13008	Marseille	France
69	Girard	Régine	4 bis rue Profess...	69008	Lyon	France
70	Yılmaz	Ekrem	Tepebaşı Bulvarı...	80050	İstanbul	Türkiye
71	Kaya	Erkan	Akay Caddesi N...	06640	Ankara	Türkiye
72	Demir	Filiz	Fethi Cadessi No...	33212	İzmir	Türkiye
73	Şahin	Ali	Sanayi Cad. No:...	23300	Elazığ	Türkiye
74	Çelik	Nur	Kazımiye Mah. O...	59860	Çorlu	Türkiye
75	佐藤	明子	東京都渋谷区...	〒150-000	東京	日本
76	鈴木	幹夫	神奈川県横浜...	〒221-083	横浜市	日本
77	田中	あずみ	大阪府大阪市...	〒534-001	大阪市	日本
78	山本	麻美	厚木市寿町3-7-...	〒243-000	厚木市	日本
79	青木	寛	川崎市川崎区...	〒210-083	川崎市	日本
80	李	伟	镜春园 75	100871	北京	中国
81	王	建国	人民大道 200	200003	上海	中国
82	张	东	武汉市汉口解...	430022	武汉	中国
83	陈	英	体育东路140-14...	510620	广州	中国
84	馬	雪	汉中门大街1号...	210029	南京	中国
85	Balázs	Ildikó	Apor Vilmos püs...	9021	Gyor	Magyarország
86	Lynesné	Antal Mária	Thököly u. 69	8660	Csabapuszta	Magyarország
87	Szabó	Gyozo	Eszéki út 28/B	7700	Mohács	Magyarország
88	Zsóri	Csilla	Arany János utc...	9764	Csempeszkopács	Magyarország
89	Zsóri	Gergely	Arany János utc...	9764	Csempeszkopács	Magyarország
90	Lyankus	István	Tas vezér u. 7	H-1113	Budapest	Magyarország
91	Kovács	Szabolcs	Külso köz	6710	Szeged	Magyarország

Figure 2: Filter and Sort with SAS Enterprise Guide

A SAS Unicode server can be used to produce ODS output. The following output formats are supported:

- HTML
- XML
- PDF
- RTF.

Examples of ODS output are shown later in the section “Generating Multilingual Reports.”

Processing Multi-byte Data

It is important to realize the differences of processing data in a single-byte versus a multi-byte environment.

UTF-8 is a variable-width multi-byte encoding in which the character codes 0x00 through 0x7F have the same meaning as ASCII. One UTF-8 character can be 1 byte, 2 bytes, 3 bytes, or even 4 bytes.

Generally, characters from alphabetic scripts (Latin, Greek, Cyrillic, Hebrew, and Arabic) are represented in either one or two bytes (for example, in the Spanish name Gómez[3] the "ó" takes up 2 bytes, the other four characters only one each). Characters from most Asian scripts are represented in three bytes. A Unicode font processes these as one character.

To process data in UTF-8, you have to use the DBCS string functions (also known as K functions). A complete list of K functions and documentation on their use can be found in the *SAS 9.2 National Language Support Reference Guide*. To use K functions properly, you need to understand the difference between byte-based offset or length and character based offset or length. Most of the K functions require character based offset or length. Under an SBCS environment, a byte-based unit is identical to a character-based unit. However, under a DBCS or MBCS environment, there are significant differences and programmers need to distinguish them. Sometimes they need to change the programming logic to use K functions. The listing of language names (in English and in the native language) below illustrates the differences in byte-based length and character-based length. Compare the 'Number of Characters' and 'Length in Bytes' columns. Language names using the Latin script might use no accented (national) characters, while for others the number of national characters is identical to the total number of characters.

Table 3: Language Names

Language Name in English	**Language Name in Native Language**	**Number of Characters**	**Length in Bytes**	**Number of National Characters**
Arabic	العربية	7	14	7
Chinese	中文(简体)	6	14	4
English	English	7	7	0
French	Français	8	9	1
Georgian	ქართული	7	21	7
German	Deutsch	7	7	0
Greek	ελληνικά	8	16	8
Hebrew	עברית	5	10	5
Hindi	हिंदी	5	15	5
Hungarian	Magyar	6	6	0
Italian	Italiano	8	8	0
Japanese	日本語	3	9	3
Korean	한국어	3	9	3

Language Name in English	Language Name in Native Language	Number of Characters	Length in Bytes	Number of National Characters
Polish	Polski	6	6	0
Portuguese	Português	9	10	1
Russian	Русский	7	14	7
Slovak	Slovenčina	10	11	1
Spanish	Español	7	8	1
Swedish	svenska	7	7	0
Thai	ไทย	3	9	3
Turkish	Türkçe	6	8	2
Vietnamese	Tiếng Việt Nam	16	20	4

The byte-based length varies according to the underlying language script. For example, the native language name for Russian is русский. The word is 7 characters in length but consists of 14 bytes. The native name for Japanese, 日本語, consists of three characters but the number of bytes is 9.

Let us examine the differences between the DBCS (K) string functions and the typical string functions. Below is an excerpt of a multilingual customer database.

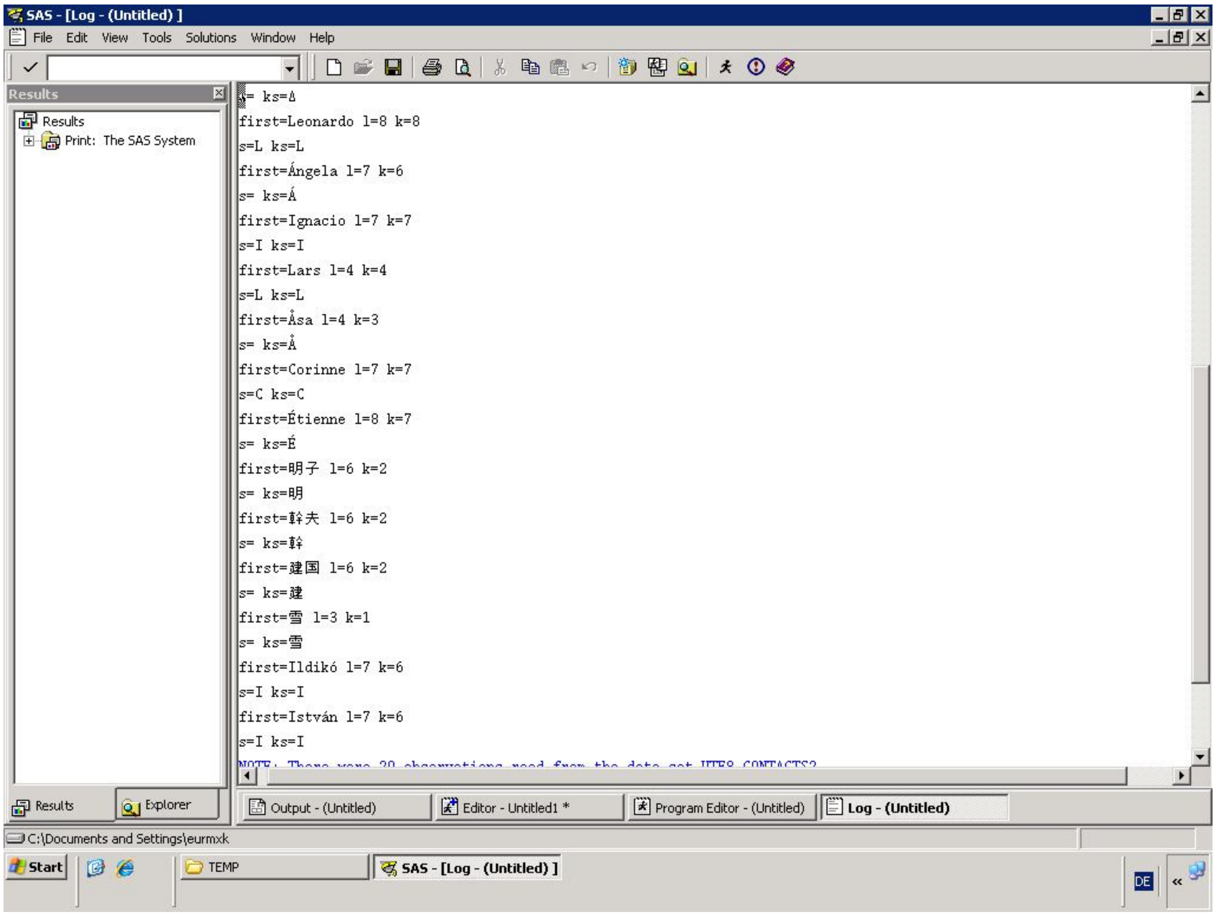
Table 4: Contact data

Name	Name (English)	First Name	First Name (English)	Street
Śmigowska	Smigowska	Świetłana	Swietlana	Ulica Marszałkowska 78/80
Perzyński	Perzynski	Piotr	Piotr	ul. Wł. Jagiełły 8
Воронин	Voronin	Борис	Boris	ул.Профессора Попова, 23
Лыжичко	Lyzhychko	Руслана	Ruslana	пер.Асбестовский 7
Παπαρίζου	Paparizou	Ζωή	Zoe	Πτολεμαίων 35
Σαββόπουλος	Savvopoulos	Κωνσταντίνος	Konstantinos	Ακτή Ποσειδώνος 48
Λάσκαρη	Laskari	Δαφνη	Dafne	Νότη Μπότσαρη 1
Martínez-Monés	Martinez-Mones	Leonardo	Leonardo	Rúa Maestranza 9
López Fernandez	Lopez Fernandez	Ángela	Angela	Avenida Padre Isla 1

Name	Name (English)	First Name	First Name (English)	Street
Nuñez Navarro	Nunez Navarro	Ignacio	Ignacio	Plaza de Castilla, 3, Piso 3
Wallin	Wallin	Lars	Lars	Smedjegatan 2
Öberg	Oberg	Åsa	Asa	Karlavägen 12
Boucher	Boucher	Corinne	Corinne	21 Place du Panthéon
Fournier	Fournier	Étienne	Etienne	91, Rue Victor Hugo
佐藤	Sato	明子	Akiko	東京都渋谷区神宮前6-29-4
鈴木	Suzuki	幹夫	Mikio	神奈川県横浜市神奈川区台町1-10　ルグラン台町3F
王	Wang	建国	Jianguo	人民大道 200
馬	Ma	雪	Xue	汉中门大街1号金鹰汉中新城21楼
Balázs	Balazs	Ildikó	Ildiko	Apor Vilmos püspök tér 3
Lyankus	Lyankus	István	Istvan	Tas vezér u. 7

Now, let us identify the first character of the first names. This process does not work with the SUBSTR function; therefore we need to use the KSUBSTR function instead. For illustrative purposes, we add the SUBSTR function, and check the number of bytes and the number of characters with the LENGTH and the KLENGTH functions.

```
data _null_;
  set utf8.contacts;
  l=length(first);
  k=klength(first);
  put first= l= k=;
  s=substr(first,1,1);
  ks=ksubstr(first,1,1);
  put s= ks=;
run;
```



```
s= ks=Á
first=Leonardo l=8 k=8
s=L ks=L
first=Ángela l=7 k=6
s= ks=Á
first=Ignacio l=7 k=7
s=I ks=I
first=Lars l=4 k=4
s=L ks=L
first=Åsa l=4 k=3
s= ks=Å
first=Corinne l=7 k=7
s=C ks=C
first=Étienne l=8 k=7
s= ks=É
first=明子 l=6 k=2
s= ks=明
first=幹夫 l=6 k=2
s= ks=幹
first=建国 l=6 k=2
s= ks=建
first=雪 l=3 k=1
s= ks=雪
first=Ildikó l=7 k=6
s=I ks=I
first=István l=7 k=6
s=I ks=I
NOTE: There were 20 observations read from the data set UTF8.CONTACTS2
```

SAS - [Log - (Untitled)]
File Edit View Tools Solutions Window Help
Results
Results
Print: The SAS System
Results
Explorer
Output - (Untitled)
Editor - Untitled1 *
Program Editor - (Untitled)
Log - (Untitled)
C:\Documents and Settings\eurmxk
Start
TEMP
SAS - [Log - (Untitled)]
DE

Figure 3: SUBSTR versus KSUBSTR

As we can see from the log, the SUBSTR function does not return any valid results unless the character is plain ASCII (as the L in the name Leonardo, for example). In this case, the results of the SUBSTR and KSUBSTR functions are identical.

We cannot assume a fixed position of multi-byte characters in the input or output, because a character in a multi-byte encoding is not equivalent to a byte (as with single-byte encodings). In a multi-byte encoding, all non-ASCII characters have a varying length, and this length varies between two and four bytes.

However, you can read in data with a fixed structure as one string and then break it up into manageable chunks with the KSUBSTR function. For example:

```
filename raw '<your raw data file>' lrecl=2048 termstr=crlf end=lastline
encoding='utf-8';
data work.utftable(drop=string reclen);
  infile raw length=reclen;
  input @;
  input string $varying2048. reclen;
  customer_id=ksubstr(string,1,10);
  customer_state=ksubstr(string,11,2);
  mail_preference=ksubstr(string,13,3);
  customer_source_name=ksubstr(string,16,10);
  birth_date=ksubstr(string,26,8);
  first_name=ksubstr(string,34,50);
  middle_name=ksubstr(string,84,50);
  last_name=ksubstr(string,134,120);
  gender_code=ksubstr(string,254,1);
  address_salutation=ksubstr(string,255,50);
  letter_salutation=ksubstr(string,305,120);
  academic_title=ksubstr(string,425,50);
  nobility_title=ksubstr(string,475,50);
  language_id=ksubstr(string,525,10);
  member_create_tms=ksubstr(string,535,19);
  member_update_tms=ksubstr(string,554,19);
  opt_out_ind=ksubstr(string,573,1);
run;
```

The following Macro Functions for NLS are available:

%KINDEX
%KLEFT and %QKLEFT
%KLENGTH
%KSCAN and %QKSCAN
%KSUBSTR and %QKSUBSTR
%KUPCASE and %QKUPCASE
%KCMPRES and %QKCMPRES

The following Autocall Macros for NLS are available:

%KLOWCASE and %QKLOWCASE
%KTRIM and %QKTRIM
%KVERIFY and %QKVERIFY

See the SAS 9.2 National Language Support Reference Guide for more detail. In all other cases we recommend that you use a %SYSFUNC call to the appropriate K function rather than using macro functions.

Example code before and after conversion:

Before: `%do %while(%length(%scan(&collist,&i)));`

After: `%do %while(%sysfunc(klength(%sysfunc(kscan(&collist,&i)))));`

Accessing and Creating Data

A Unicode server session is ideal to process data from various sources. Data can be read into the session from three sources:

- External files
- SAS libraries
- DBMS tables.

External Files

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 UTF-8, the SAS session encoding of our Unicode Server.

When a file contains only character data and its encoding is different from the SAS session encoding , use the ENCODING= option on the FILENAME, INFILE, or FILE statement to tell SAS what the file encoding is so SAS can transcode the data from its original encoding to the current SAS session encoding; for example:

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

See the *SAS 9.2 National Language Support Reference Guide* for details on using the ENCODING= option.

When an external file contains a mix of character and binary data, then you must use the KCVT function to convert individual fields from the file encoding to the session encoding. See the *SAS 9.2 National Language Support Reference Guide* for details on using the KCVT function.

SAS Libraries

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

You need to increase variable lengths to prevent truncation during *transcoding* of the data to UTF-8. To write in English, you need 7-bit ASCII characters (this includes the uppercase and lowercase letters A through Z and a through z, the digits 0 through 9, and various special characters such as . / ? , * # and so on). In UTF-8, they are 1 byte. For many European languages, the impact is limited because only national characters like ç, ñ, and é use 2 bytes. However, if you convert Arabic, Cyrillic, or Japanese data to UTF-8, the data takes more bytes. Make sure your variables are large enough to store the additional bytes.

The following examples show you how to import SAS 9 or SAS 8 data. In either case, you need to use the CVP (character variable padding) engine to expand character variable lengths so that the character data does not truncate.

```
/*********************************************************************/
/* SAS 9 data example
*/
/*
*/
```

```
/*******************************************************************/

libname myfiles 'SAS data-library';
libname expand cvp 'SAS data-library';
data myfiles.utf8;4
set expand.wlatin1;
run;
```

Omitting the CVP engine results in the following transcoding error:

```
ERROR: Some character data was lost during transcoding in the dataset
EXPAND.WLATIN1.
NOTE: The data step has been abnormally terminated.
NOTE: The SAS System stopped processing this step because of errors.
```

This action prevents the risk of data corruption or data loss.

```
/*********************************************************************/
/* SAS 8 data example
*/
/*
*/
/*******************************************************************/

libname myfiles 'SAS data-library';
libname expand cvp 'SAS data-library'
data myfiles.utf8;
set expand.wlatin1 (encoding=wlatin1);
run;
```

The only difference in the SAS 8 and SAS 9 code example is that the SAS 8 data set needs the ENCODING= option while the SAS 9 data set does not. Leaving out the CVP engine in the SAS 8 example would result in the same transcoding error above.
Note: Omitting the ENCODING= option does not provoke an error message, but the output data might be corrupted.

The CVP engine does not adjust formats or informats automatically. So if you have attached formats to your data variables, you might still run into truncation issues. In this case, you need to adjust these formats manually. The macro code below allows you to estimate the lengths of character variables (and format widths) when migrating data from a legacy encoding to UTF-8.

```
%macro utf8_estimate(dsn,vars=_character_,maxvarlen=200,maxvars=20,out=);
data _utf8_est_ ;
set &dsn end=eof ;
array _c_ $ &vars ;
retain _maxvlen_1-_maxvlen_&maxvars ;
array _mv_ _maxvlen_1-_maxvlen_&maxvars ;
length temp $ &maxvarlen ;
length _vname_ $ 40 ;

do i = 1 to dim(_c_);
temp = put(_c_[i], $utf8x&maxvarlen..) ;
utf8len = length( temp ) ;
_mv_[i] = max(_mv_[i], utf8len ) ;
```

```
end ;

if eof then do ;
%if %length(&out) > 0 %then %do ;
call execute("data &out(encoding=utf8) ;");
%end ;
do i = 1 to dim(_c_) ;
_vname_ = vname(_c_[i]) ;
_vlen_ = vlength(_c_[i]) ;
_vlenopt_ = _mv_[i] ;
_vlenmax_ = max(_mv_[i], _vlen_) ;
if _vlenopt_ > _vlen_ then _vconv_ = 'Y' ;
else _vconv_ = 'N' ;
output ;
%if %length(&out) > 0 %then %do ;
if _vconv_ = 'Y' then do ;
call execute ('length ' || _vname_ || '$' || put(_vlenopt_,best12.) || ';'
) ;
end;
%end ;
end;
%if %length(&out) > 0 %then %do ;
call execute("set &dsn ; ");
call execute("run; ");
%end ;
end ;
label _vname_ = 'Variable Name'
_vlen_ = 'Current Variable Length'
_vlenopt_ = 'Estimated Variable Length'
_vlenmax_ = 'Required Variable Length'
_vconv_ = 'Conversion Required?' ;
keep _vname_ _vlen_ _vlenopt_ _vlenmax_ _vconv_ ;
run;
%if %length(&out) = 0 %then %do ;
proc print data=_utf8_est_ label ;
run;
%end ;
%mend ;
```

Expected input is the data set you want to investigate; the input data set should be in the current SAS session encoding. The output data set needs to be specified after the out= macro keyword; the data is converted to UTF-8 after estimating the required character variable lengths. The macro code also creates a temporary data set _utf8_est_ with information about 'Current Variable Length', 'Estimated Variable Length' and 'Required Variable Length') and if there was a 'Conversion Required'. If an adjustment was necessary the current and estimated/required lengths differ, if not they are the same.

Let has have a look at an example. Imagine you have a data set similar to SASHELP.CLASS but with data in Russian:

Наблюдения	ИМЯ	ПОЛ	ВОЗРАСТ	РОСТ (Дюйм)	ВЕС (Фунт)	ПОЛ	ГРУППА
1	Алексей	M	14	69.0	112.5	мальчик	взрослый (large)
2	Алиса	F	13	56.5	84.0	девочка	подросток (middle)

Наблюдения	ИМЯ	ПОЛ	ВОЗРАСТ	РОСТ (Дюйм)	ВЕС (Фунт)	ПОЛ	ГРУППА
3	Барбара	F	13	65.3	98.0	девочка	подросток (middle)
4	Катерина	F	14	62.8	102.5	девочка	взрослый (large)
5	Геннадий	M	14	63.5	102.5	мальчик	взрослый (large)
6	Евгений	M	12	57.3	83.0	мальчик	ребенок (small)
7	Мария	F	12	59.8	84.5	девочка	ребенок (small)
8	Жанна	F	15	62.5	112.5	девочка	взрослый (large)
9	Андрей	M	13	62.5	84.0	мальчик	подросток (middle)
10	Антон	M	12	59.0	99.5	мальчик	ребенок (small)
11	Даша	F	11	51.3	50.5	девочка	ребенок (small)
12	Людмила	F	14	64.3	90.0	девочка	взрослый (large)
13	Луиза	F	12	56.3	77.0	девочка	ребенок (small)
14	Мария	F	15	66.5	112.0	девочка	взрослый (large)
15	Филлип	M	16	72.0	150.0	мальчик	взрослый (large)
16	Роберт	M	12	64.8	128.0	мальчик	ребенок (small)
17	Иван	M	15	67.0	133.0	мальчик	взрослый (large)
18	Роман	M	11	57.5	85.0	мальчик	ребенок (small)
19	Юрий	M	15	66.5	112.0	мальчик	взрослый (large)

A printing of the temporary data set _utf8_est_ looks like this:

Наблюдения	Variable Name	Current Variable Length	Estimated Variable Length	Required Variable Length	Conversion Required?
1	name	12	16	16	Y
2	Sex	1	1	1	N
3	sexl	7	14	14	Y
4	group	19	27	27	Y

This means the lengths for the variables name, sexl and group was increased (in case of sexl the length was doubled). The picture might be different with Western European data, and again different with data from Asian languages.

DBMS Tables

SAS/ACCESS® software enables SAS to share data with Oracle, DB2, and other relational database management systems (DBMS). SAS can read and write data to a database management system (DBMS) in exactly the same way as reading from or writing to a SAS library.

When accessing data in many DBMS from a SAS session, it is critical that the DBMS client software which is used to access the DBMS server knows what the SAS session encoding is. The DBMS client software is responsible for either transcoding the data to the encoding expected by the DBMS or telling DBMS how the data is encoded in the client application so that the server can do the transcoding. Each DBMS client software application has its own way of being configured with the client application encoding. In a SAS Unicode server you must be sure to configure the DBMS client software to specify that the SAS application is using the UTF8 encoding. The specific configuration methods of each DBMS client are discussed in the sections that follow, and in the Appendix. In addition, a trouble shooting section at the end of this chapter discusses what can happen if the DBMS client is not properly configured for the UTF8 Session encoding in a Unicode server.

SAS/ACCESS Interface to Oracle

The SAS/ACCESS LIBNAME statement below associates the libref ora with an Oracle database management system. If the Oracle DBMS contains a table named contacts, you can reference ora.contacts as if it were a SAS data set.

```
libname ora oracle
  user=tester password=testpw
  path=oradb;
```

You can create a new Oracle table called gercon by copying a SAS data set with German contact data by using DATA step programming, for example:

```
data ora.gercon;
set utf8.gercon;
run;
```

For database clients, you must set certain parameters. In Oracle, the NLS_LANG parameter is responsible for setting the *locale* used by the client application and the database server. NLS_LANG is used to let Oracle know what *character set* your client application is using so that Oracle can convert from the client's character set to the database character set, when needed.

It consists of three components, which are specified in the following format, including the punctuation:

NLS_LANG=<Language>_<Territory>.<character set>

For our purposes, we can disregard the first two components and concentrate on the <character set> part of NLS_LANG. In a Unicode Server, this part of NLS_LANG must be UTF8 to match the SAS session encoding. See the Trouble Shooting section to understand what happens if NLS_LANG uses some other character set (or encoding[5]) value in a Unicode Server.

NLS_LANG is set as a local environment variable on UNIX platforms. It is set in the registry or in the environment on Windows platforms.

How SAS/ACCESS interface to Oracle Interprets Length in Oracle

In Oracle 9i, the NLS_LENGTH_SEMANTICS parameter determines whether a new column of character data type is calculated in bytes or in characters. Calculating column lengths in bytes is called *byte semantics*, measuring the lengths in characters is called *character semantics*. By default, CHAR and VARCHAR2 columns are measured in bytes, NCHAR, and NVARCHAR2 Oracle columns in characters.

However, SAS **always measures the length** of its variables **in bytes**, and in 9.1.3 SAS/ACCESS simply doubled the Oracle variable length. For example, Oracle has an NCHAR column with length of 10 characters (say Chinese, which should use 3 bytes per character in UTF-8 encoding), when transferred to SAS via SAS/ACCESS, SAS 9.1.3 simply doubled the NCHAR(10), which is now 10*2=20 bytes in SAS. But it really should be 10*3=30 bytes. So the last 10 (30 – 20) bytes are truncated.

In SAS 9.2, several new features were added to make the variable length more flexible. Four new LIBNAME options support this flexibility:

DBCLIENT_MAX_BYTES - The maximum number of bytes per character in the database client encoding (matches with SAS encoding). This option is used as the multiplying factor for adjusting the column lengths for CHAR/NCHAR columns for the client encoding. DEFAULT: Always set to match the maximum bytes per character of SAS session encoding. This default should be sufficient for most cases, and so there should be no need to set this option. For example, with a SAS session encoding of UTF-8, the value is 4.

DBSERVER_MAX_BYTES - This is the maximum number of bytes per character in the database server encoding. This option is used to derive the number of character values from the lengths for columns created with byte semantics. Use this option to adjust the client lengths of columns created with byte semantics. DEFAULT: Usually 1 but might be set to another value if the information is available from the Oracle server.

ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS - This option indicates whether the lengths should be adjusted for CHAR/VARCHAR data type lengths specified using BYTE semantics. When set to NO, any lengths specified with BYTE semantics on the server are used as on the client side. When set to YES, the lengths are divided with the DBSERVER_MAX_BYTES value and then multiplied with the DBCLIENT_MAX_BYTES value. DEFAULT: YES if DBCLIENT_MAX_BYTES is greater than 1; NO if DBCLIENT_MAX_BYTES is equal to 1.

ADJUST_NCHAR_COLUMN_LENGTHS - This option indicates whether the lengths should be adjusted for NCHAR/NVARCHAR data type columns. When set to NO, any lengths specified with NCHAR/NVARCHAR columns are multiplied with the maximum bytes per character value of the database national character set. When this option is set to YES, the lengths are multiplied with the DBCLIENT_MAX_BYTES value.

This process can be illustrated in the following table:

Table 5: LENGTH in Oracle

Data type	Adjust_nchar_column_lengths		Adjust_byte_semantic_column_lengths	
	YES(default)	NO	YES(default)	NO
NCHAR NVARCHAR	VL=CL[6] dbclient_max_bytes	VL=CL national_max_bytes		

CHAR		VL=/dbserver_max_bytes*dbclient_max_bytes	VL=CL
VARCHAR			
CHAR(n char)		VL=CL* dbclient_max_bytes	

In our example, with an NCHAR (10) column in Oracle, the option ADJUST_NCHAR_COLUMN_LENGTHS set to "YES", and DBCLIENT_MAX_BYTES to 3, the length after expansion is VL=CL*dbclient_max_bytes =10 * 3 =30 bytes. This means, there is no truncation and no extra byte length from the UTF8 default DBCLIENT_MAX_BYTES of 4.

Trouble Shooting SAS/ACCESS Encoding Settings

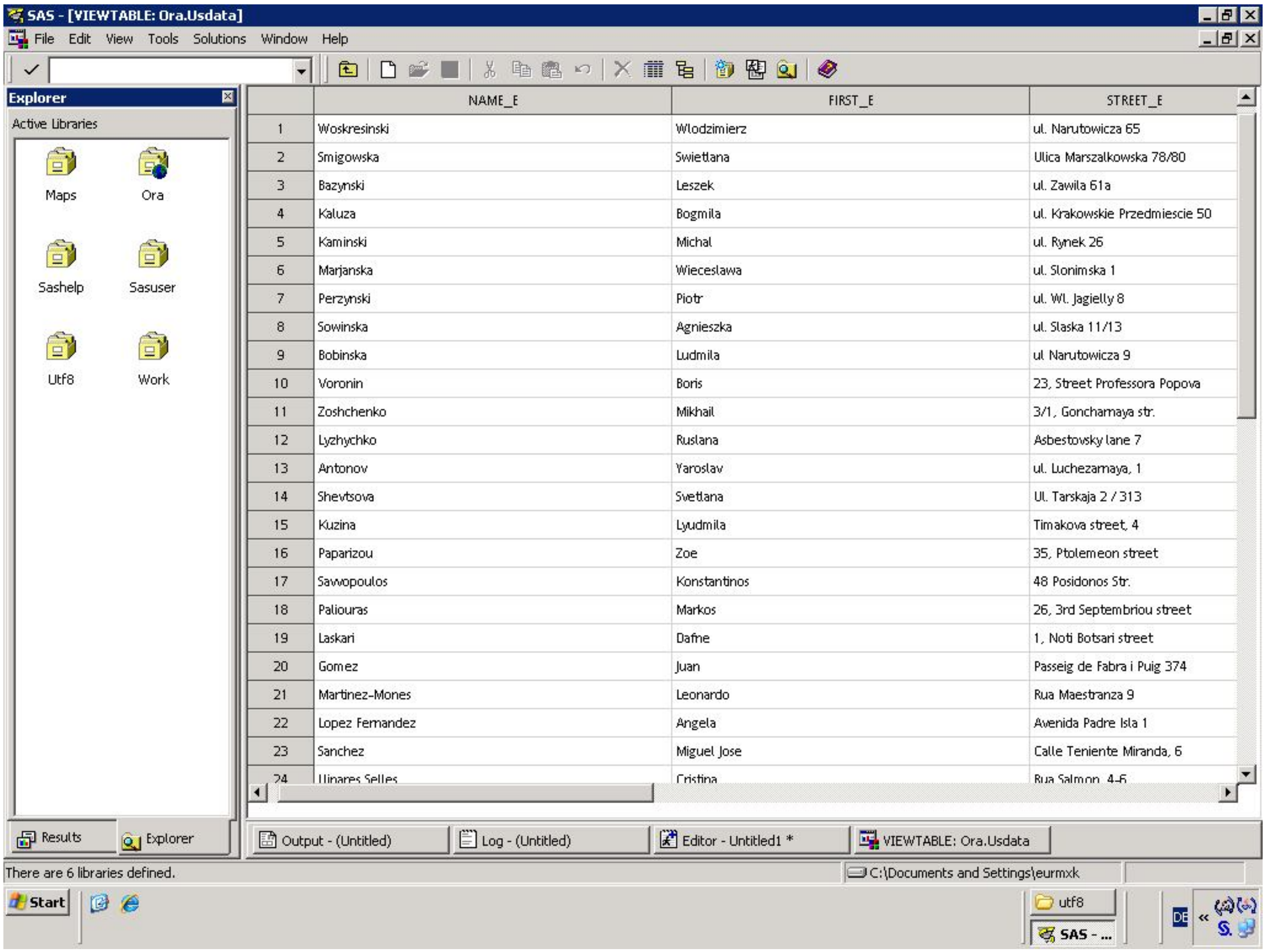
The previous sections have explained how to configure DBMS clients that need to know that the SAS session encoding is UTF8. This section gives an example of what you might experience if that setting has not been correctly made. Then we also consider an example where the setting has been made to UTF8, but instead of starting a Unicode server with a UTF8 session encoding, you use a SAS session with some other session encoding. We use Oracle for these examples but the same results would happen with DB2, Informix, or Teradata if they were configured incorrectly (see: Appendix A).

Example 1 – DBMS Client incorrectly configured

Let us examine what happens if the DBMS client encoding is set to Wlatin1 in a SAS Unicode server. In Oracle this would happen if the character set component of the NLS_LANG setting were WE8MSWIN1252. So here we are assuming that, the Oracle database had been created with NLS_CHARACTERSET set to UTF8, the SAS session encoding is UTF-8, and NLS_LANG was set incorrectly to AMERICAN_AMERICA.WE8MSWIN1252.

If we execute the following program where utf8.usdata contains only text characters as defined in US_ASCII (7-bit ASCII) the program works because all these UTF8 character codes are identical to their ASCII values.

```
data ora.usdata;
set utf8.usdata;
run;
```



	NAME_E	FIRST_E	STREET_E
1	Woskresinski	Wlodzimierz	ul. Narutowicza 65
2	Smigowska	Swietlana	Ulica Marszalkowska 78/80
3	Bazynski	Leszek	ul. Zawila 61a
4	Kaluza	Bogmila	ul. Krakowskie Przedmiescie 50
5	Kaminski	Michal	ul. Rynek 26
6	Marjanska	Wieceslawa	ul. Slonimska 1
7	Perzynski	Piotr	ul. Wl. Jagielly 8
8	Sowinska	Agnieszka	ul. Slaska 11/13
9	Bobinska	Ludmila	ul Narutowicza 9
10	Voronin	Boris	23, Street Professora Popova
11	Zoshchenko	Mikhail	3/1, Goncharnaya str.
12	Lyzhychko	Ruslana	Asbestovsky lane 7
13	Antonov	Yaroslav	ul. Luchezarnaya, 1
14	Shevtsova	Svetlana	Ul. Tarskaja 2 / 313
15	Kuzina	Lyudmila	Timakova street, 4
16	Paparizou	Zoe	35, Ptolemeon street
17	Savvopoulos	Konstantinos	48 Posidonos Str.
18	Paliouras	Markos	26, 3rd Septembriou street
19	Laskari	Dafne	1, Noti Botsari street
20	Gomez	Juan	Passeig de Fabra i Puig 374
21	Martinez-Mones	Leonardo	Rua Maestranza 9
22	Lopez Fernandez	Angela	Avenida Padre Isla 1
23	Sanchez	Miguel Jose	Calle Teniente Miranda, 6
24	Linares Selles	Cristina	Rua Salmon, 4-6

There are 6 libraries defined.

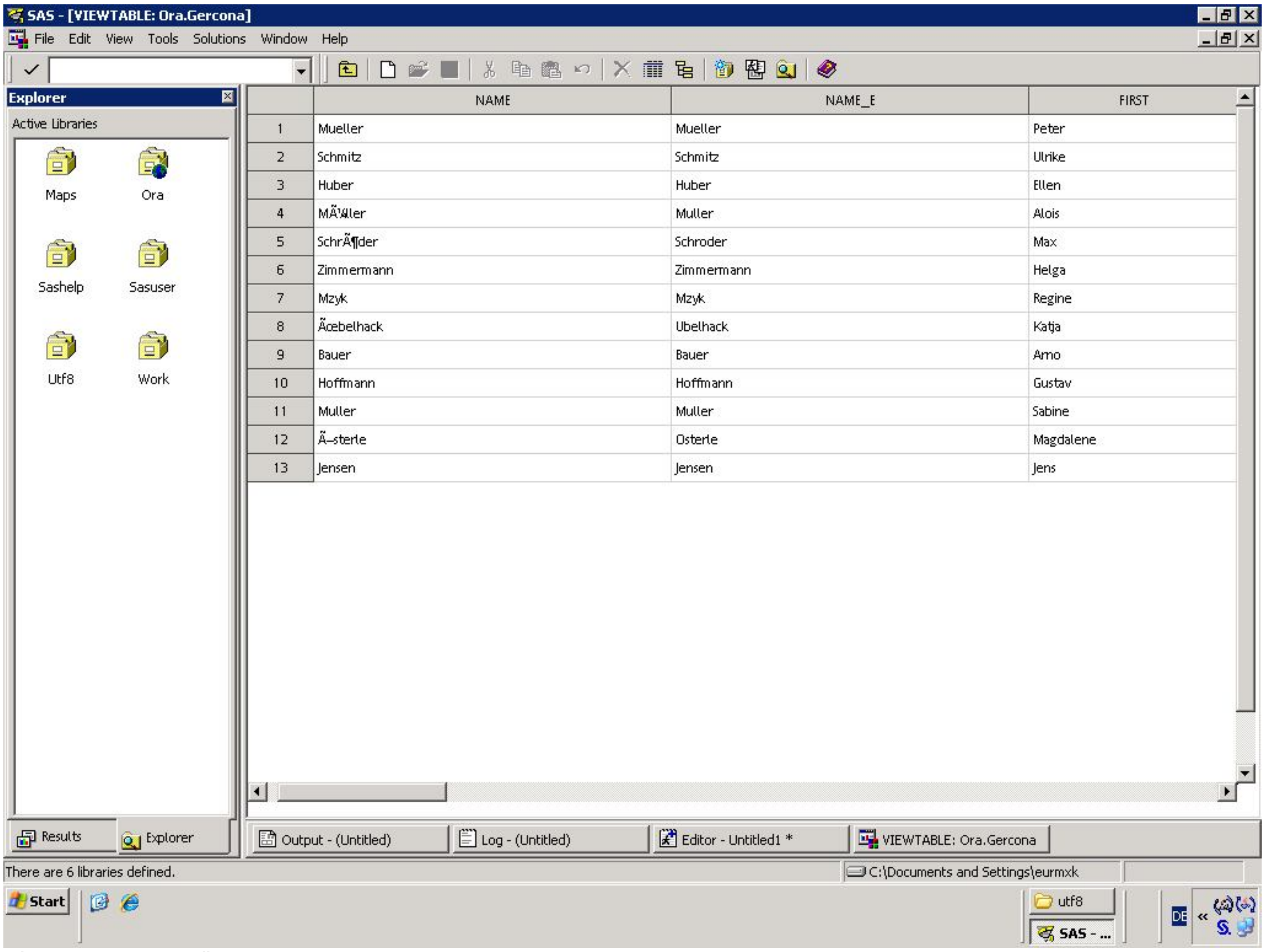
C:\Documents and Settings\eurmxk

Figure 4: DBMS Client incorrectly configured

However if we run the program with utf8.gercon that contains German national characters in the text data:

```
data ora.gercona;
set utf8.gercon;
run;
```

Oracle interprets each byte of the multibyte national characters as individual wlatin1 character and transcodes each byte to its equivalent UTF8 character. The data stored in Oracle is not correctly UTF8 encoded but if you access it from the same or another SAS Unicode session with the NLS_LANG set to AMERICAN_AMERICA.WE8MSWIN1252, you get the data back that looks correct. Oracle has just reversed the previously incorrect transcoding. However, if you access that Oracle data from any other correctly configured environment including a SAS Unicode session with a correctly configured NLS_LANG of AMERICAN_AMERICA.UTF8, you do not get the data back as the erroneous transcoding is not reversed.


SAS - [VIEWTABLE: Ora.Gercona]

	NAME	NAME_E	FIRST
1	Mueller	Mueller	Peter
2	Schmitz	Schmitz	Ulrike
3	Huber	Huber	Ellen
4	MÃ¼ller	Muller	Alois
5	SchrÃ¶der	Schroder	Max
6	Zimmermann	Zimmermann	Helga
7	Mzyk	Mzyk	Regine
8	Ãœbelhack	Ubelhack	Katja
9	Bauer	Bauer	Arno
10	Hoffmann	Hoffmann	Gustav
11	Muller	Muller	Sabine
12	Ã–sterle	Osterle	Magdalene
13	Jensen	Jensen	Jens

There are 6 libraries defined.

Figure 5: DBMS Client

To see what was really stored, you need to check the numeric values of the characters in the database (as explained further below).

Example 2 – SAS session incorrectly configured

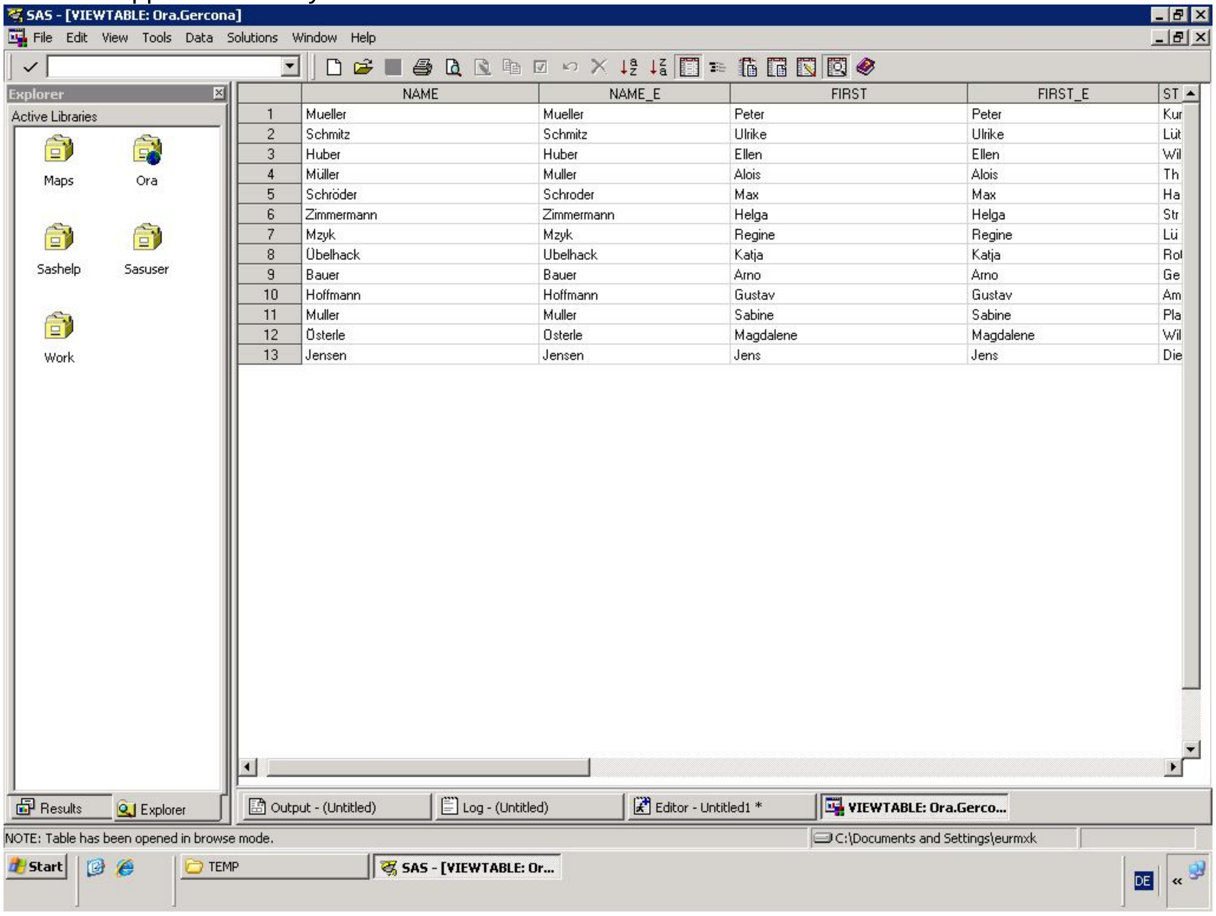
Now let us examine what happens if the DBMS client encoding is set to UTF8 but you use a SAS Session not in the UTF8 encoding to access it. Here we are assuming that the Oracle database had been created with NLS_CHARACTERSET set to UTF8, the SAS session encoding is Wlatin1, and NLS_LANG was set incorrectly to AMERICAN_AMERICA. UTF8.

Nevertheless, we can erroneously update a UTF8 Oracle database from a SAS session using wlatin1:

```
data ora.gercona;
set wlatin1.gercon;
run;
```

NLS_LANG is the same as the database character set. Oracle assumes that the data uses the same encoding, and does not convert it. The German contact data is stored as data that has been delivered by the SAS client, in wlatin1.

The data appears correctly when accessed from the SAS wlatin1 session:



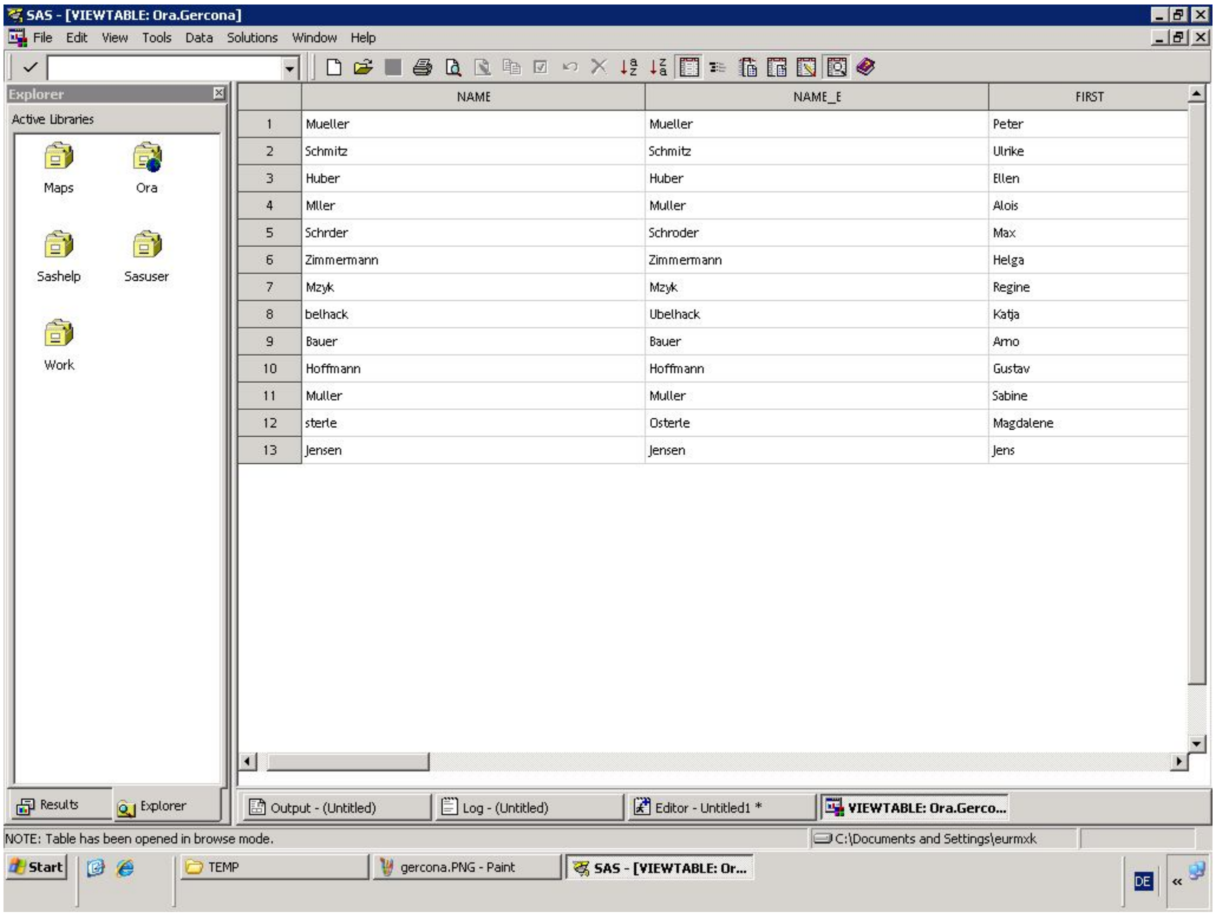
SAS - [VIEWTABLE: Ora.Gercona]

	NAME	NAME_E	FIRST	FIRST_E	ST
1	Mueller	Mueller	Peter	Peter	Kur
2	Schmitz	Schmitz	Ulrike	Ulrike	Lüt
3	Huber	Huber	Ellen	Ellen	Wil
4	Müller	Muller	Alois	Alois	Th
5	Schröder	Schroder	Max	Max	Ha
6	Zimmermann	Zimmermann	Helga	Helga	Str
7	Mzyk	Mzyk	Regine	Regine	Lü
8	Übelhack	Ubelhack	Katja	Katja	Rot
9	Bauer	Bauer	Arno	Arno	Ge
10	Hoffmann	Hoffmann	Gustav	Gustav	Am
11	Muller	Muller	Sabine	Sabine	Pla
12	Österle	Osterle	Magdalene	Magdalene	Wil
13	Jensen	Jensen	Jens	Jens	Die

NOTE: Table has been opened in browse mode.

Figure 6: SAS session incorrectly configured

However, when accessed from a UTF-8 session, the German characters do not display at all:

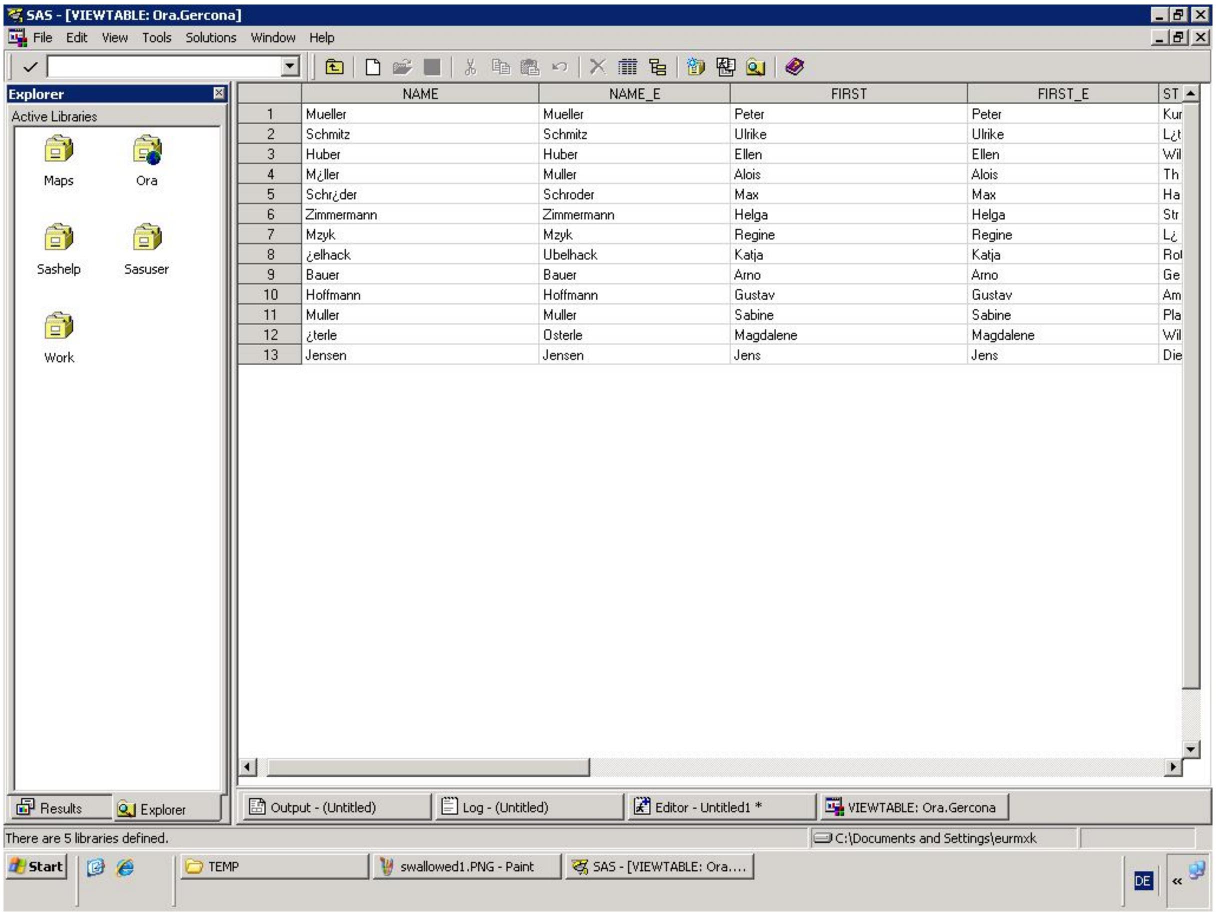


	NAME	NAME_E	FIRST
1	Mueller	Mueller	Peter
2	Schmitz	Schmitz	Ulrike
3	Huber	Huber	Ellen
4	Mller	Muller	Alois
5	Schrder	Schroder	Max
6	Zimmermann	Zimmermann	Helga
7	Mzyk	Mzyk	Regine
8	belhack	Ubelhack	Katja
9	Bauer	Bauer	Arno
10	Hoffmann	Hoffmann	Gustav
11	Muller	Muller	Sabine
12	sterle	Osterle	Magdalene
13	Jensen	Jensen	Jens

NOTE: Table has been opened in browse mode.

Figure 7: SAS session incorrectly configured

Let us now see what happens if we set the NLS_LANG to AMERICAN_AMERICA.WE8MSWIN1252. In a wlatin1 SAS session, we see inverted question marks instead of the German characters:



	NAME	NAME_E	FIRST	FIRST_E	ST
1	Mueller	Mueller	Peter	Peter	Kur
2	Schmitz	Schmitz	Ulrike	Ulrike	L¿t
3	Huber	Huber	Ellen	Ellen	Wil
4	M¿ller	Muller	Alois	Alois	Th
5	Schr¿der	Schroder	Max	Max	Ha
6	Zimmermann	Zimmermann	Helga	Helga	Str
7	Mzyk	Mzyk	Regine	Regine	L¿
8	¿elhack	Ubelhack	Katja	Katja	Rol
9	Bauer	Bauer	Arno	Arno	Ge
10	Hoffmann	Hoffmann	Gustav	Gustav	Am
11	Muller	Muller	Sabine	Sabine	Pla
12	¿terle	Osterle	Magdalene	Magdalene	Wil
13	Jensen	Jensen	Jens	Jens	Die

Figure 8: SAS session incorrectly configured

The inverted question mark is a replacement character that indicates that something has gone wrong during a character conversion.

The problem is that we have incorrect data in the database. To check what was stored in the database, we can use the dump command in SQL*Plus:

```
SQL> SELECT DUMP(name,1616)FROM gercona;

DUMP(NAME,1616)
-------------------------------------------------------
Typ=1 Len=7 CharacterSet=UTF8: M,u,e,l,l,e,r
Typ=1 Len=7 CharacterSet=UTF8: S,c,h,m,i,t,z
Typ=1 Len=5 CharacterSet=UTF8: H,u,b,e,r
Typ=1 Len=6 CharacterSet=UTF8: M,fc,l,l,e,r
Typ=1 Len=8 CharacterSet=UTF8: S,c,h,r,f6,d,e,r
Typ=1 Len=10 CharacterSet=UTF8: Z,i,m,m,e,r,m,a,n,n
Typ=1 Len=4 CharacterSet=UTF8: M,z,y,k
Typ=1 Len=8 CharacterSet=UTF8: dc,b,e,l,h,a,c,k
Typ=1 Len=5 CharacterSet=UTF8: B,a,u,e,r
Typ=1 Len=8 CharacterSet=UTF8: H,o,f,f,m,a,n,n
Typ=1 Len=6 CharacterSet=UTF8: M,u,l,l,e,r
```

```
DUMP(NAME,1616)
-------------------------------------------------------
Typ=1 Len=7 CharacterSet=UTF8: d6,s,t,e,r,l,e
Typ=1 Len=6 CharacterSet=UTF8: J,e,n,s,e,n

13 rows selected.
```

In this example, we have chosen to display character values in hexadecimal format. ASCII characters are printed if they correspond to printable ASCII codes. This greatly improves the readability of the output. The name Österle has been stored with the Ö as 0xd6, which is the wlatin1 code for this letter. However, the database character set has been defined as UTF-8. This means that the German characters have been stored in the wrong encoding.

We must repeat the action. With NLS_LANG set to AMERICAN_AMERICA.WE8MSWIN1252, we can run our SAS wlatin1 session again and create another Oracle table:

```
data ora.gerconb;
set wlatin1.gercon;
run;
```

This time Oracle coverts the German data from the client's character set (wlatin1) to the database character set (UTF-8); and the data is stored correctly in the database.

We can now check what was stored in the database. We can use the dump command again but this time from our SAS session:

```
proc sql;
  connect to oracle (user=tester pw=testpw path=oradb);
  select * from connection to oracle
     (select dump(name,1616), name from gerconb);
quit;
```

The output looks like this:

```
                                                 The SAS System

DUMP(NAME,1616)                                        NAME

---------------------------------------------------------------------------------------------------

Typ=1 Len=7 CharacterSet=UTF8: M,u,e,l,l,e,r           Mueller

Typ=1 Len=7 CharacterSet=UTF8: S,c,h,m,i,t,z           Schmitz

Typ=1 Len=5 CharacterSet=UTF8: H,u,b,e,r               Huber

Typ=1 Len=7 CharacterSet=UTF8: M,c3,bc,l,l,e,r         Müller

Typ=1 Len=9 CharacterSet=UTF8: S,c,h,r,c3,b6,d,e,r     Schröder

Typ=1 Len=10 CharacterSet=UTF8: Z,i,m,m,e,r,m,a,n,n    Zimmermann

Typ=1 Len=4 CharacterSet=UTF8: M,z,y,k                 Mzyk

Typ=1 Len=9 CharacterSet=UTF8: c3,9c,b,e,l,h,a,c,k     Übelhack

Typ=1 Len=5 CharacterSet=UTF8: B,a,u,e,r               Bauer
```

```
Typ=1 Len=8 CharacterSet=UTF8: H,o,f,f,m,a,n,n             Hoffmann

Typ=1 Len=6 CharacterSet=UTF8: M,u,l,l,e,r                 Muller

Typ=1 Len=8 CharacterSet=UTF8: c3,96,s,t,e,r,l,e           Österle

Typ=1 Len=6 CharacterSet=UTF8: J,e,n,s,e,n                 Jensen
```

Now the name Österle, has been stored with the Ö in 2 bytes as 0xc396, which is the UTF-8 code for this letter.

The process is documented in the following table:

Table 6: NLS_LANG and SAS session encoding

DB character set	NLS_LANG character set	SAS session encoding	Characters stored in DB as
UTF-8	WE8MSWIN1252	UTF-8	Undefined
UTF-8	UTF-8	UTF-8	UTF-8
UTF-8	UTF-8	Wlatin1	Wlatin1
UTF-8	WE8MSWIN1252	Wlatin1	UTF-8

This process is a simple example to illustrate what can go wrong if you do not carefully specify the respective parameters. The database should be created with a character set that comprises characters from all the languages you need to support. With a global set of languages, UTF-8 is your best choice. The NLS_LANG setting on the client should then match your SAS session encoding.

For more information about Oracle NLS parameters, refer to the *Oracle Database Globalization Support Guide*.

Multilingual Collation

A universal encoding allows you to handle multilingual data without changing the encoding based on the target language. However, the encoding does not tell you where one language ends and where another begins. The encoding does not help you order the data.

By default, the names in our contacts database are sorted according to the session encoding, which is UTF-8. You can also explicitly set the SORTSEQ option to Unicode (UTF-8):

```
proc sort data=utf8.contacts SORTSEQ=”Unicode (UTF-8)”;
   by name;
run;
```

The result is the same. Using this option does not produce a collating sequence that is culturally correct because the option value only orders characters according to their position in this particular encoding.

Using SORTSEQ=LINGUISTIC, however, produces a correct sort. Alternatively, you can specify SORTSEQ=UCA. If you need to order multilingual data, but the context does not define a particular locale, the Unicode Collation Algorithm (UCA) provides a convenient way to put the data in sequence.

In a multilingual environment, the relative ordering of scripts has been defined by UCA as (using this example) Latin – Greek – Cyrillic – CJK (Chinese – Korean – Japanese).

More information about linguistic collation can be obtained from the SAS technical paper *Linguistic Collation: Everyone Can Get What They Expect - Sensible Sorting for Global Business Success.*

Language Switching

SAS products have been localized, or translated into, since SAS 6. *Localization* of these products involves a translation of message files, ODS templates, the SAS registry, and other files. The language used for messages displayed by SAS is determined at start up by settings in the SAS configuration file. With SAS 9.2. support was added to the Unicode server so that the language of certain texts matches the SAS LOCALE option when localization is available.

This feature, referred to as *Language Switching,* is controlled by the LOCALELANGCHG option. If LOCALELANGCHG is on, the language switching feature is enabled and the setting of the LOCALE= option determines the language used for procedure output and user interface elements. ODS fonts used by a product or procedure also match the LOCALE. Messages written to the SAS log, including notes, warnings, and errors, always display in the language used at SAS system start up.

More information about language switching can be obtained from the SAS technical paper *Changing Language during a SAS Session.* Currently, language switching is officially supported only in the Unicode server.

Generating Multilingual Reports

In the code sample below, PROC FREQ creates a table of frequencies and percentages for the country variable of our contacts database. We then create output in RTF format:

```
ods _all_ close;
libname utf8 'c:\utf8';
ods rtf file='c:\temp\contacts.rtf';
proc freq data=utf8.contacts;
 table country;
run;
ods rtf close;
```

Table 7: Multilingual output with ODS RTF

Country				
Country	**Frequency**	**Percent**	**Cumulative Frequency**	**Cumulative Percent**
Danmark	8	8.79	8	8.79
Deutschland	13	14.29	21	23.08
España	9	9.89	30	32.97
France	10	10.99	40	43.96
Magyarország	7	7.69	47	51.65
Polska	9	9.89	56	61.54
Sverige	10	10.99	66	72.53
Türkiye	5	5.49	71	78.02
Ελλάδα	4	4.40	75	82.42
Россия	6	6.59	81	89.01
中国	5	5.49	86	94.51
日本	5	5.49	91	100.00

The code below creates a pie chart with PROC GCHART:

```
goptions reset=all;
libname utf8 'c:\utf8';
ods html file='c:\temp\contacts_graph.html';
goptions device=activex;
proc gchart data=utf8.contacts;
pie3d country;
run;
quit;
```

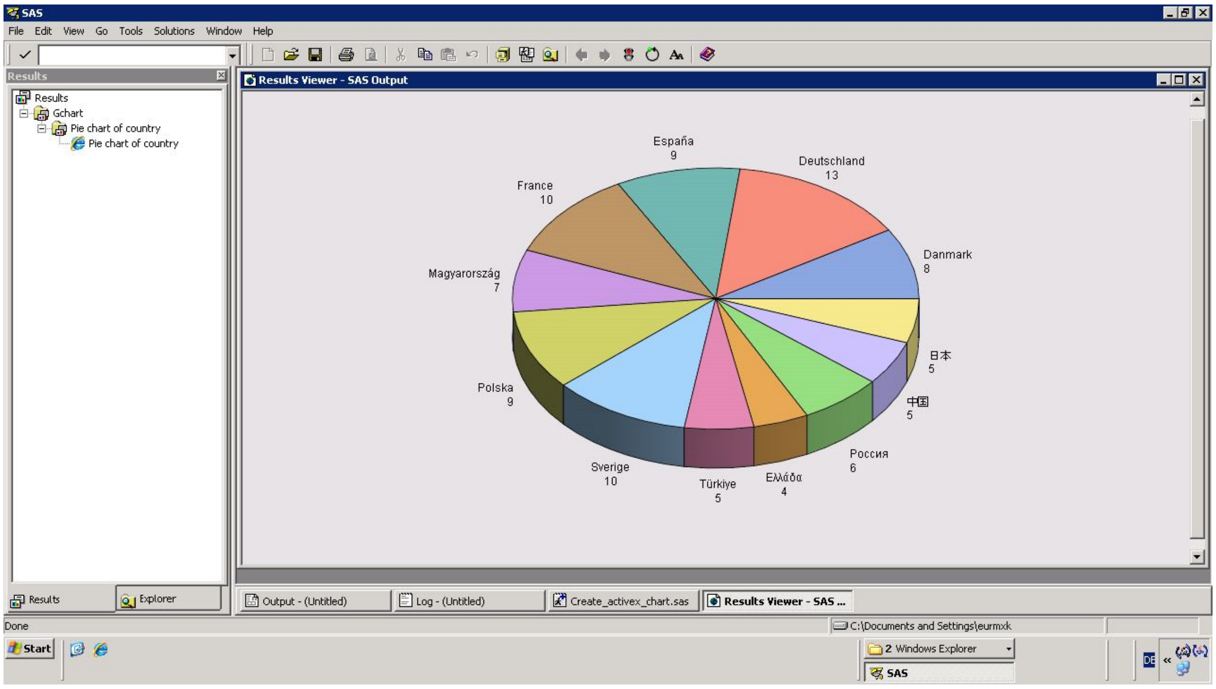


Figure 9: Multilingual output with PROC GCHART

Best Practices

To make the most efficient use of a SAS server, you should store the data in Unicode format with an encoding of UTF-8. By default, when a file is created, it inherits the current session encoding. One of your first steps in converting an application to run with the SAS Unicode server is to convert the legacy data files.

Use the CVP engine described previously to convert your files and avoid truncation problems. CVP is an engine that must be specified in the LIBNAME statement on the input data so truncation does not occur during transcoding. By specifying the CVP engine, the default expansion is 1.5 times the current variable length. There are options to specifically set the expansion. See SAS Help for information about CVPMultiplier, CVPBytes, and CVPEngine. By default, PROC COPY tries to create an output file with the same attributes as the input file. Therefore, you must specify the NOCLONE option when using PROC COPY to convert legacy data.

Using PROC COPY re-creates indexes, but the data needs to be allocated with the CVP engine, not the SAS/SHARE® server. Allocating data on a SAS/SHARE server with CVP has issues if the data has indexes.

Once the data is converted, it can be updated only in a UTF-8 session. Non-UTF-8 sessions can read the data but not write to the data.

Restrictions

There are a few restrictions to the SAS Unicode server:

1. You cannot run a SAS Unicode server on z/OS or OpenVMS.
2. Full-screen non-Java components such as SAS/AF® software and SAS/FSP® software do not work properly with multi-byte data. Therefore, products that rely on full screen capability are not supported. Various other SAS visual products based on display manager technology do not work either.
3. OLEDB local data providers do not fully support multi-lingual data.
4. PDF and RTF output created with ODS does not render BiDi[7] text in the correct order. However, you can use the ODS tagsets.rtf destination to create BIDI output, as in the following example:

   ```
   ods tagsets.rtf file="test.rtf" OPTIONS(TROWD="\rtlrow");
   proc print data=utf8.contacts; run;
   ods tagsets.rtf close;
   ```

Conclusion

With the SAS Unicode server, you can handle a global set of languages in a single application. It allows you to meet your business needs maintaining and processing multilingual data in one SAS session.

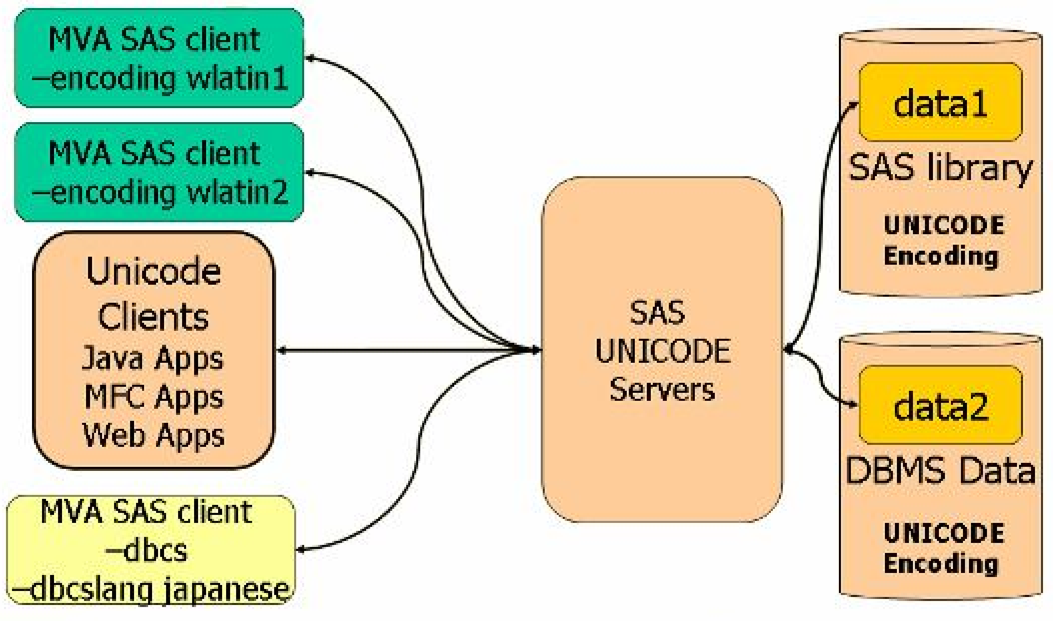


Figure 10: Support for Global Applications

Acknowledgments

The author wishes to thank the following people (listed in alphabetical order) who have taken time to review and contribute to this paper: Junichi Arai, Elizabeth Bales, Sandy Brown, Betty Joyner, Shinichiro Kayano, Joji Kobayashi, Philip Shelton, Jack Wallace, Kansun Xia and Ray (Runfeng) Zhang. He is indebted to Jin Luo and Nancy (Na) Zhou for the information about the SAS/ACCESS Interfaces contained in Appendix A. Thank you to Dan Harrell, Kathy Restivo and Rachel Wynne for providing editorial assistance.

References

Informix Software, Inc. 1997. Informix Guide to GLS Functionality, Version 9.1 (Part No. 000-4818, March 1997).

International Business Machines Corporation 2006. National Language Support Guide and Reference, SC10-4380-00.

NCR Corporation 2006. Teradata Database. International Character Set Support. Release V2R6.2. (B035-1125-096K, September 2006).

Oracle Corporation 2002. Oracle9i Database Globalization Support Guide, Release 2 (9.2), Part No. A96529-01.

Progress Software Corporation 2009. DataDirect Connect® Series for ODBC - Reference. Release 6.0. March 2009.

SAS Institute Inc. 2008. SAS/ACCESS® 9.2 for Relational Databases: Reference. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2008. SAS® 9.2 National Language Support (NLS): Reference Guide. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2008. SAS Institute technical paper. “Linguistic Collation: Everyone Can Get What they Expect - Sensible Sorting for Global Business Success."

SAS Institute Inc. 2008. SAS Institute technical paper. "SAS 9.1.3 Service Pack 4 in a Unicode Environment."

SAS Institute Inc. 2009. SAS Institute technical paper. “Changing language during a SAS session."

Appendix

Appendix A: Configuring SAS/ACCESS Interfaces for UTF-8

SAS/ACCESS® Interface to DB2

To send and receive Unicode data, the DB2 client must use the DB2CODEPAGE value of 1208; this is the client *code page*[5] for UTF-8.

To see whether your DB2CODEPAGE is already set, look in your environment variables, or check SAS using the following code:

```
data _null_;
x=sysget('DB2CODEPAGE');
put x=;
run;
```

If you plan to exchange data between databases with different code pages, check the *DB2 Administration Guide* to determine whether the code pages you have are compatible.

SAS/ACCESS® Interface to Informix

The SAS/ACCESS interface to Informix works on UNIX. On Windows, you need to use the SAS/ACCESS interface to ODBC. The environment variables to set are CLIENT_LOCALE and DB_LOCALE.
For example, you might want to connect to an Informix database that has stored data in the Chinese encoding of GB 18030. The Informix wrapper has set Informix environment variables to: CLIENT_LOCALE=zh_cn.UTF8 GL_USEGLU=1. Now, you must add the following setting to your db2dj.ini file so that the Informix client correctly converts the GB 18030 data to Unicode: DB_LOCALE=zh_cn.GB18030-2000.
Refer to the *Informix Guide to GLS Functionality* for more information about code page conversions.

SAS/ACCESS® Interface to Microsoft SQL Server

The SAS/ACCESS Interface for Microsoft SQL Server is an out of box solution that provides connectivity between SAS on UNIX Platforms and Microsoft SQL Server databases for data access and update. The interface communicates through a Microsoft SQL Server ODBC driver from DataDirect Technologies. Conceptually, there is a database code page and an application code page, and the driver converts between them.

On Windows platforms SAS/ACCESS to ODBC needs to be used. Here, the application code page is determined by the regional setting of the Windows OS. On UNIX, the application code page is set by the IANAAppCodePage connection string attribute, which is part of the ODBC.ini file. The ODBC.ini file located in !sasroot/misc/dbi/ and looks something like:

```
[SQLServer]
Driver=/home/data/SAS/SASFoundation/9.2/misc/dbi/lib/S0msss22.so
Description=DataDirect 5.2 SQL Server Wire Protocol
Address=xxxxxxx
AlternateServers=
AnsiNPW=Yes
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=test2
LoadBalancing=0
LogonID=xxxxxx
Password=xxxxxx
QuotedId=yes
ReportCodePageConversionErrors=0
IANAAppCodePage=4
```

For information about the IANAAppCodePage values, see the appropriate chapter of the DataDirect Connect® Series for ODBC Reference Guide.

SQL Server supplies four different data types that can hold textual data: CHAR, VARCHAR, NCHAR, and NVARCHAR. Data is stored as Unicode in NCHAR/NVARCHAR columns in SQL Server no matter what collation (database codepage) was set.

SAS/ACCESS to ODBC on Windows provides an option “auto translation” to control the transcoding of CHAR data. “Auto translation” can be set in ODBC data source configuration dialog box. It is checked on by default.

For the **CHAR** data types in SQL server, only when the “auto translation “option is on and the client code page is different from SQL server database collation, there will be a transcoding from one side to another.

For the **NCHAR** data types in SQL server, the data transcoding process depends on the SAS session encoding. Because in SAS 9.2 there is better Unicode support, UTF-16 data in the NCHAR type is transcoded into UTF-8 directly. But for other encodings, data transcoding depends on code page of client operating system.

Before SAS accesses data from SQL server, you should make sure the data was correctly stored in the database. You can use the ***CAST*** function for this purpose either in Microsoft SQL Server Management Studio or in SAS with the SQL Pass-Through Facility with something like the following code:

```
proc sql;
   connect to SQLSVR as mydb
      (datasrc=SQLServer user = sasuser pass = XXXXXXXXXXX);

   select * from connection to mydb
      (select col2, cast(col2 as binary) col2 from test);
quit;
```

Variables might have to be expanded as shown below:

SQL server Data Type	SAS Variable Length
CHAR (char, varchar2)	No expansion
NCHAR (nchar, nvarchar2)	For Unicode session Column length * ***3***
	For non-Unicode session Column length * **2**

In SAS9.2, if you launch a Unicode Session to process multilingual data in SQL server through ODBC, NCHAR is the recommended data type to be used to store data in SQL server. If you have to access data in CHAR type in SQL server, make sure set the code page in the operating system according to the SAS Session encoding.

SAS/ACCESS® Interface to MySQL

SAS/ACCESS Interface to MySQL is standard access technology that provides a LIBNAME engine interface to MySQL data. MySQL is an open source database software provider.

MySQL data is stored using the UTF-8 encoding. By default, the SAS MYSQL LIBNAME engine assumes that the Detail Data Store (DDS) data is stored using Latin1 encoding, and it transcodes Latin1 to UTF-8. However, if the DDS has a different character set encoding (for example, UTF-8), then errors occur. In that case, you must modify the MySQL libnames as follows.

1. Log on to SAS Management Console as an administrator.
2. Open the Solutions repository.
3. Under Data Library Manager SAS Libraries, right-click on SASSDM and select Properties.
4. Click the Options tab and then the Advanced Options button.
5. In Advanced Options, click the Connection tab.
6. In the User-defined connection initialization command field, enter the
7. following string:
   dbconinit="set names 'charsetname';"

For example, for UTF-8, you would enter
dbconinit="set names ’UTF-8’;"

When the Detail Data Store (DDS) is installed, the encoding that is used to create the data sets is taken from the SAS session. Therefore, it is important to install SAS with the desired session encoding specified before configuring the solutions (which installs the DDS).
When you install DDS sample data, the sample data sets replace the previously installed data sets. Because the DDS sample data is created and shipped with Latin1 encodings, the resulting DDS is always Latin1. If the SAS session encoding is not Latin1, this mismatch can lead to errors.
If you have installed SAS with a session encoding of UTF-8, take these steps to modify the sample data encoding:

1. Install the sample data, as described in the System Administration Guide.
2. Run a SAS job that calls the %CNVECODE macro for the DDS and StageDDS libraries:

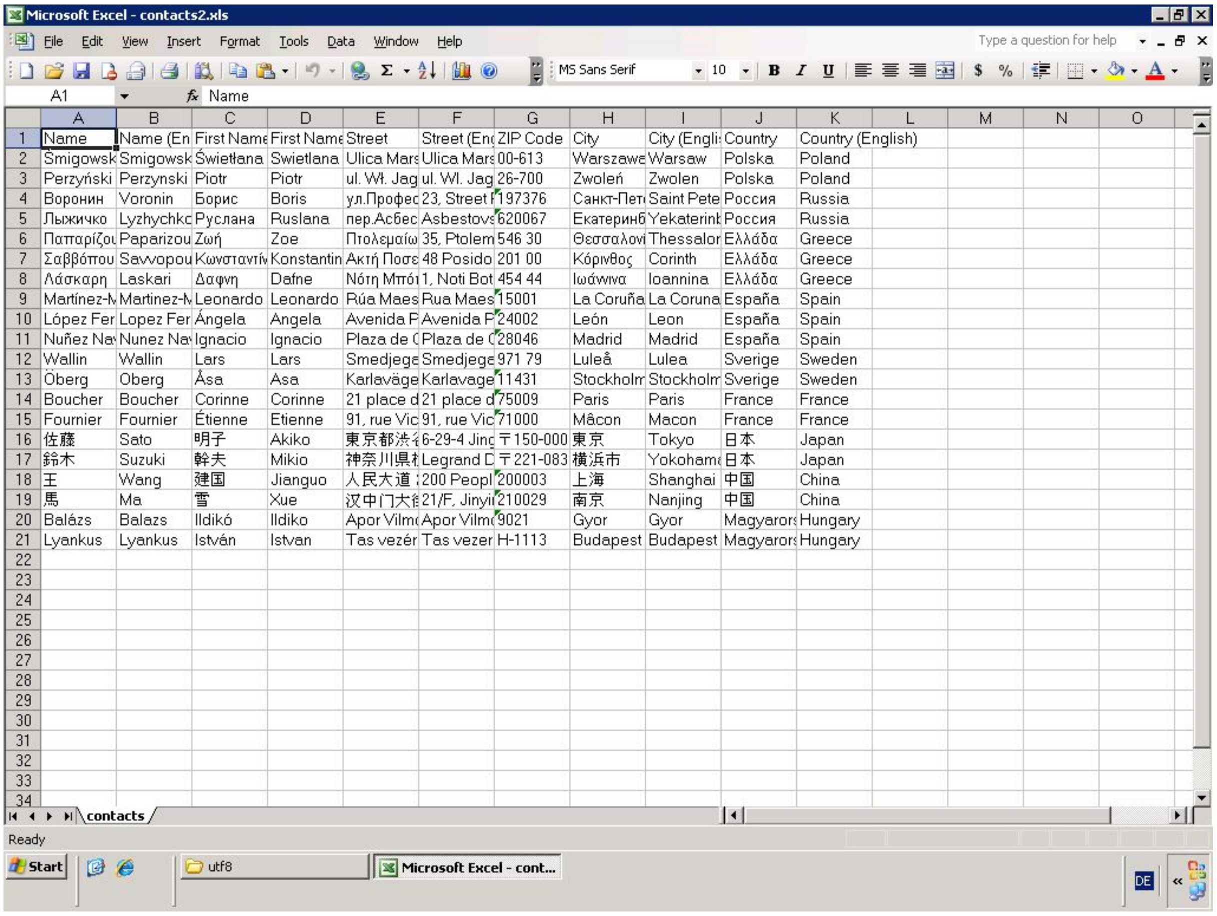
```
%cnvecode(ses_libname=DDS, ses_encoding=charsetname);
%cnvecode(ses_libname=StageDDS, ses_encoding=charsetname);
```

For example, to change the character set encoding to utf-8, you would use these calls:

```
%cnvecode(ses_libname=DDS, ses_encoding=’utf-8’);
%cnvecode(ses_libname=StageDDS, ses_encoding=’utf-8’);
```

The encoding specified should correspond to the encoding for your SAS session (typically specified in the SASV9.cfg configuration file). Do not use ASCIIANY or an encoding that differs from the SAS session encoding. For a list of valid charsetname values, see the list under “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” in the SAS National Language Support (NLS): User’s Guide.

SAS/ACCESS® to PC Files



	A	B	C	D	E	F	G	H	I	J	K
1	Name	Name (En	First Name	First Name	Street	Street (En	ZIP Code	City	City (Engli	Country	Country (English)
2	Śmigowsk	Smigowsk	Świetłana	Swietlana	Ulica Mars	Ulica Mars	00-613	Warszawa	Warsaw	Polska	Poland
3	Perzyński	Perzynski	Piotr	Piotr	ul. Wł. Jag	ul. Wl. Jag	26-700	Zwoleń	Zwolen	Polska	Poland
4	Воронин	Voronin	Борис	Boris	ул.Профес	23, Street I	197376	Санкт-Пет	Saint Pete	Россия	Russia
5	Лыжичко	Lyzhychko	Руслана	Ruslana	пер.Асбес	Asbestovs	620067	Екатеринб	Yekaterink	Россия	Russia
6	Παπαρίζου	Paparizou	Ζωή	Zoe	Πτολεμαίω	35, Ptolem	546 30	Θεσσαλονί	Thessalon	Ελλάδα	Greece
7	Σαββόπου	Savvopou	Κωνσταντίν	Konstantin	Ακτή Ποσε	48 Posido	201 00	Κόρινθος	Corinth	Ελλάδα	Greece
8	Λάσκαρη	Laskari	Δαφνη	Dafne	Νότη Μπότ	1, Noti Bot	454 44	Ιωάννινα	Ioannina	Ελλάδα	Greece
9	Martínez-M	Martinez-M	Leonardo	Leonardo	Rúa Maes	Rua Maes	15001	La Coruña	La Coruna	España	Spain
10	López Fer	Lopez Fer	Ángela	Angela	Avenida P	Avenida P	24002	León	Leon	España	Spain
11	Nuñez Na	Nunez Na	Ignacio	Ignacio	Plaza de (	Plaza de (	28046	Madrid	Madrid	España	Spain
12	Wallin	Wallin	Lars	Lars	Smedjega	Smedjega	971 79	Luleå	Lulea	Sverige	Sweden
13	Öberg	Oberg	Åsa	Asa	Karlaväge	Karlavage	11431	Stockholm	Stockholm	Sverige	Sweden
14	Boucher	Boucher	Corinne	Corinne	21 place d	21 place d	75009	Paris	Paris	France	France
15	Fournier	Fournier	Étienne	Etienne	91, rue Vic	91, rue Vic	71000	Mâcon	Macon	France	France
16	佐藤	Sato	明子	Akiko	東京都渋	6-29-4 Jing	〒150-000	東京	Tokyo	日本	Japan
17	鈴木	Suzuki	幹夫	Mikio	神奈川県	Legrand D	〒221-083	横浜市	Yokohama	日本	Japan
18	王	Wang	建国	Jianguo	人民大道	200 Peopl	200003	上海	Shanghai	中国	China
19	馬	Ma	雪	Xue	汉中门大	21/F, Jinyi	210029	南京	Nanjing	中国	China
20	Balázs	Balazs	Ildikó	Ildiko	Apor Vilm	Apor Vilm	9021	Gyor	Gyor	Magyaror	Hungary
21	Lyankus	Lyankus	István	Istvan	Tas vezér	Tas vezer	H-1113	Budapest	Budapest	Magyaror	Hungary

Figure 11: Spreadsheet with multilingual address data

In SAS 9.2, a libname option, UNICODE=YES | NO, is available for SAS/ACCESS, Excel, and PCFILES LIBNAME engines.

If you specify UNICODE=YES, SAS binds all text data in wide-character format and transcodes character data to Unicode data that can be in a Microsoft Access file or in an Excel file. If you are running SAS in a Windows UTF-8 session, SAS/ACCESS and Excel LIBNAME engines automatically set UNICODE=YES. However, you have to specify this LIBNAME option when you use PCFILES LIBNAME engine.

The following code imports multilingual data from a spreadsheet into a UTF-8 data set:

```
proc import out= work.contacts
  datafile= "c:\utf8\contacts.xls"
  dbms=excel replace;
  range="contacts";
  getnames=yes;
  mixed=no;
  scantext=yes;
  usedate=yes;
  scantime=yes;
```

```
run;
```

This only works if you have file-system-level access to the Excel file.

If you need to access the same spreadsheet from a UNIX machine, you need a LIBNAME statement using the PCFILES engine. An example would be:

```
   libname myxls pcfiles server="ipaddress" type=excel port=8621
         path="c:\utf8\contacts.xls" unicode=yes;
```

You do not need to specify “type=excel”. If the file extension is xls (or xlsb, xlsx, xlsm for Excel 2007), type=excel is automatically assumed. The same goes for .mdb and .accdb extensions. They assume type=access. Also, if the default port=8621 is used, the port option might be omitted. However, if the PC Files server is started with a different, non-standard port number, then the PORT option must be specified.

Please make sure that

- PC Files Server is installed on the 32-bit Windows system, and the files you want to read from are also located here, or there is file-system-level network access with something like PATH=\\server\my_dir\mysheet.xls. PC Files Server can also run on a 64-bit Windows system, but it will run in 32-bit compatibility mode.
- SAS/ACCESS interface to PC files is installed on the UNIX/LINUX system.
- PC Files Server must be started before you can access files located in Windows PC from UNIX/LINUX.
- UNIX/LINUX users must know the following points before they can access Windows files:
    - server name, which means the Windows short name.
    - service port used by the PC Files Server, you can change the port as your wish in the PC Files Server, and after change, restart the PC Files Server to make it take effect.
    - exact file path you want to access.

SAS/ACCESS® Interface to Teradata

The Teradata database converts the server character set to the session character set. The session character set is associated with each session and specified by the application.

To check the Teradata database Server Character set info, you can run the command:
SHOW TABLE DBC.HOSTS
To check the client character set support, you can run the command:
SELECT CharSetName,InstallFlag FROM dbc.CharTranslations.

To specify UTF-8 as the client character set in the Teradata session, modify the clispb.dat file to include the following entries:

```
charset_type=N
charset_id=UTF8
```

Set the COPLIB environment variable to point to the location of the clispb.dat file.

If your SAS session is not using UTF-8 but a legacy encoding such as GB2312 for Simplified Chinese on UNIX or Windows, add the following to clispb.dat:

```
charset_type=N
charset_id=SCHGB2312_1T0
```

For other charset_id values, check the *Teradata Database International Character Set Support* manual for more information.

Even if the client parameters have been set correctly, you might still experience problems when data has been stored in a wrong encoding or has been corrupted otherwise. To check what was stored in the database, you can use the CHAR2HEXINT function. For more information about CHAR2HEXINT, see the *Teradata® RDBMS SQL Reference: Functions and Operators* manual.

You might also encounter data truncation if the number of bytes the Teradata database exports exceeds the number of bytes the client application expects to receive. In Teradata, the mechanism for exporting the appropriate number of bytes for a client application is the export width. System-wide export width tables define the export width of characters during export to a client. The values used for the export width at run time are based on:

Active export width table

Server character set

Client character set used by the application

At the system level, there are there export width table can be use to determine export widths. it is described in the table below.

Export Width Table Name	**Export-ID**	**Description**
Expected Default	0	Provides reasonable default export widths for server character set and client form-of-use. This is the initial default table and uses the same export width values as were provided for V2R3.
Compatibility Default	1	Preserves functional compatibility with releases before V2R3
Maximum Default	2	Provides maximum default export widths for server character set and client form-of-use.

In Teradata the character data type can be stored either as character or as byte, which depends on the server character set, as detailed in the following table:

Server CharacterSet	**Char type if defined in terms of…**	**The maximum value for n**
LATIN	Characters	64000
UNICODE	Characters	32000
GRAPHIC	Characters	32000
KANJI1	byte	64000
KANJISIJIS	byte	32000

The following table4 illustrates the number of bytes exported from the various server character sets to the various client character sets for the Expected Default export width table (Export Width Table ID = 0).

Client Character set	**Server Character set**	**The number of bytes exported For a CHARACTER(n) Column**
Any single-byte character set	◆ LATIN ◆ UNICODE ◆ KANJISIJIS	n
KanjiEUC	◆ LATIN ◆ KANJISIJIS ◆ KANJI1	n
	◆ UNICODE	2n
KanjiEBCDIC	◆ LATIN ◆ KANJISIJIS ◆ KANJI1	n
	◆ UNICODE	2n+2
KanjiSJIS	◆ LATIN ◆ KANJISIJIS ◆ KANJI1	n
	◆ UNICODE	2n
UTF8	◆ LATIN	2n
	◆ UNICODE	3n
	◆ KANJISJIS	n
	◆ KANJI1	
UTF16	◆ LATIN	2n
	◆ UNICODE	
	◆ KANJISJIS	
	◆ KANJI1	
◆ HANGULKSC5601_2R4 ◆ SCHGB2312_1T0 ◆ SDHANGULKSC5601_4R4 ◆ SDTCHBIG5_3R0 ◆ SDSCHGB2312_2T0 ◆ TCHBIG5_1R0	◆ LATIN ◆ KANJISJIS ◆ KANJI1	n
	◆ UNICODE	2n
HANGULEBCDIC933_1II SCHEBCDIC935_21J SDHANGULEBCDIC933_5II SDSCHEBCDIC935_6IJ SDTCHEBCDIC937_7IB TCHEBCDIC937_3IB	◆ LATIN ◆ KANJISJIS ◆ KANJI1	n
	◆ UNICODE	2n+2

Appendix B: Configuring BI servers as Unicode servers

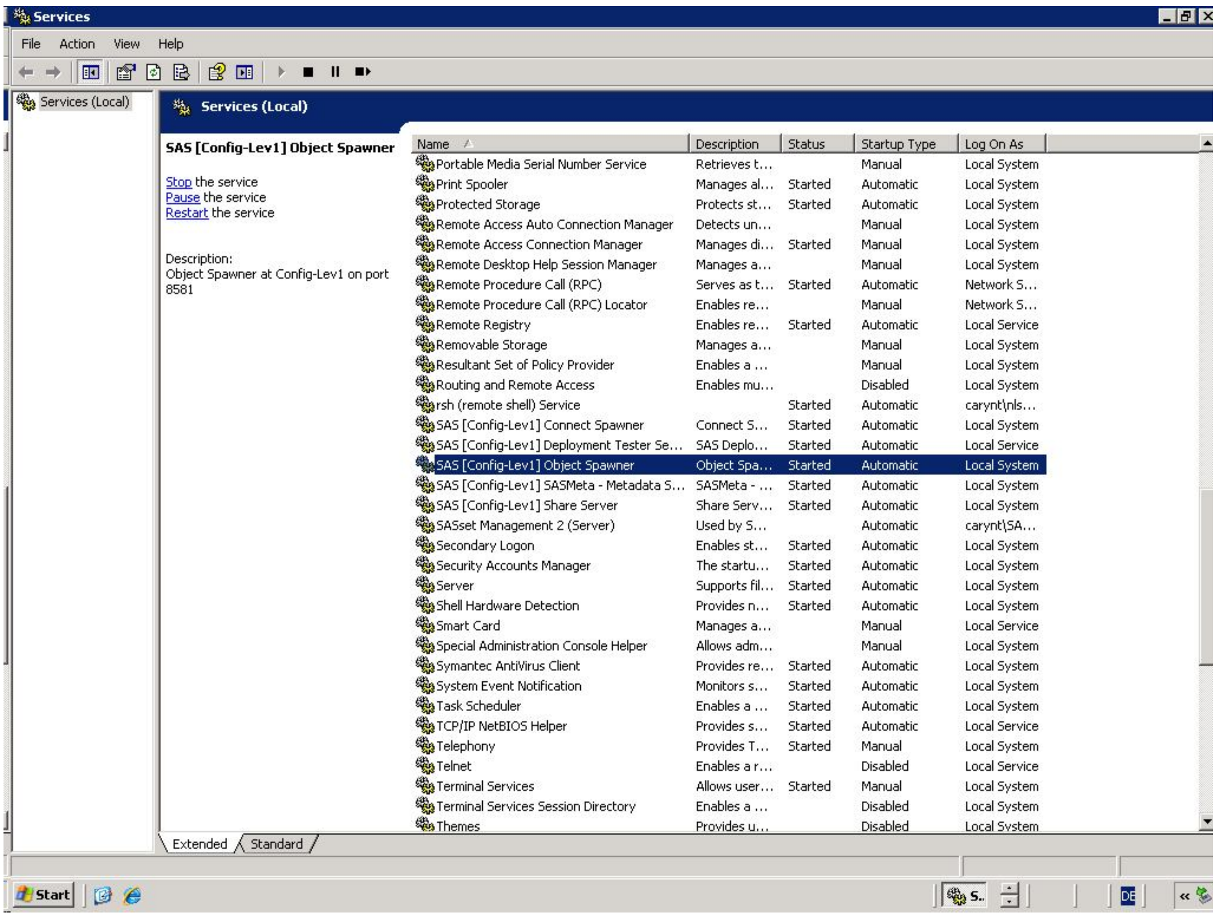
To configure a SAS® BI Server environment do the following:

1. You should have installed an English SAS version as default. If the language is not English, language-specific metadata are created in SAS Foundation like SAS servers, SAS users, SAS system folders, and so forth. If some of the metadata objects are localized in SAS Foundation repository, this may result in unpredictable errors.
2. Stop all SAS servers.
3. Configure your servers and the SAS® Management Console.
4. Restart the metadata server.
5. Restart the SAS® Management Console.
6. Set up fonts for SAS® Web Report Studio.

We shall now take you through each of these steps again in more detail. The examples below are often Windows-specific but UNIX users will also find material geared to their platforms.

Stop SAS servers

Select Start ► Setting ►Control panel ►Administrative Tools ►Services. Select SAS [Config-Lev1] Object Spawner and stop the service by right clicking on the name of the service and choose stop. Likewise for the other SAS servers. You need to stop the Metadata server the last.



SAS [Config-Lev1] Object Spawner

Stop the service
Pause the service
Restart the service

Description:
Object Spawner at Config-Lev1 on port 8581

Name	Description	Status	Startup Type	Log On As
Portable Media Serial Number Service	Retrieves t...		Manual	Local System
Print Spooler	Manages al...	Started	Automatic	Local System
Protected Storage	Protects st...	Started	Automatic	Local System
Remote Access Auto Connection Manager	Detects un...		Manual	Local System
Remote Access Connection Manager	Manages di...	Started	Manual	Local System
Remote Desktop Help Session Manager	Manages a...		Manual	Local System
Remote Procedure Call (RPC)	Serves as t...	Started	Automatic	Network S...
Remote Procedure Call (RPC) Locator	Enables re...		Manual	Network S...
Remote Registry	Enables re...	Started	Automatic	Local Service
Removable Storage	Manages a...		Manual	Local System
Resultant Set of Policy Provider	Enables a ...		Manual	Local System
Routing and Remote Access	Enables mu...		Disabled	Local System
rsh (remote shell) Service		Started	Automatic	carynt\nls...
SAS [Config-Lev1] Connect Spawner	Connect S...	Started	Automatic	Local System
SAS [Config-Lev1] Deployment Tester Se...	SAS Deplo...	Started	Automatic	Local Service
SAS [Config-Lev1] Object Spawner	Object Spa...	Started	Automatic	Local System
SAS [Config-Lev1] SASMeta - Metadata S...	SASMeta - ...	Started	Automatic	Local System
SAS [Config-Lev1] Share Server	Share Serv...	Started	Automatic	Local System
SASset Management 2 (Server)	Used by S...		Automatic	carynt\SA...
Secondary Logon	Enables st...	Started	Automatic	Local System
Security Accounts Manager	The startu...	Started	Automatic	Local System
Server	Supports fil...	Started	Automatic	Local System
Shell Hardware Detection	Provides n...	Started	Automatic	Local System
Smart Card	Manages a...		Manual	Local Service
Special Administration Console Helper	Allows adm...		Manual	Local System
Symantec AntiVirus Client	Provides re...	Started	Automatic	Local System
System Event Notification	Monitors s...	Started	Automatic	Local System
Task Scheduler	Enables a ...	Started	Automatic	Local System
TCP/IP NetBIOS Helper	Provides s...	Started	Automatic	Local Service
Telephony	Provides T...	Started	Manual	Local System
Telnet	Enables a r...		Disabled	Local Service
Terminal Services	Allows user...	Started	Manual	Local System
Terminal Services Session Directory	Enables a ...		Disabled	Local System
Themes	Provides u...		Disabled	Local System

On UNIX environments all SAS servers can be started/stopped by running the shell script: <Configuration directory>/Lev1/sas.servers {start|stop|restart|status}

(Re-)Configure servers

Since all BI servers refer to the file '!SASROOT\SASV9.CFG' you just need to modify it as below:

-CONFIG "C:\Program Files\SASFoundation\9.2\nls\u8\SASV9.CFG"

In SAS9.2 the SASV9.CFG file in directory'!SASROOT\SASFoundation\9.2\nls\u8' contains the following lines:

```
-DBCS
-ENCODING UTF-8
```

When you invoke SAS with the SASV9.CFG file in directory '!SASROOT\SASFoundation\9.2\nls\u8', you get an English version of SAS with Unicode support.

In order to set up the Unicode server on UNIX, you need to manually re-create a symbolic link from !sasroot/sas to bin/sas_u8.

```
cd /usr/local/SAS/SASFoundation/9.2
rm sas
ln -s bin/sas_u8 sas
```

Restart the metadata server

Select Start ►Settings ► Control panel ►Administrative Tools ►Services. Select SAS [Config-Lev1] SASMeta – Metadata Server and start the service by right clicking on the name of the service and choose start.

Restart the SAS® Management Console

The **SAS® Management Console** needs to be invoked under the English mode to avoid registering localized strings into metadata. Administrative information should be independent of a specific language.

You can set the locale for the SAS® Management Console to English by using the *Locale Setup Manager*. The Locale Setup Manager is a Java client that enables users to configure the locale of SAS Java clients. SAS Locale Setup Manager updates your Java application's .ini file to the locale you choose. For example, it adds the following lines to the sasmc.ini file when you select en_US:

JavaArgs_14=-Duser.language=en

JavaArgs_15=-Duser.country=US

After the modification, select Start ►Programs ► SAS ► SAS Management Console to restart the SAS® Management Console.

Set up fonts for SAS® Web Report Studio

SAS® Web Report Studio comes with four files: `ServerFonts.xml`, `ClientFonts.xml`, `LocalServerFonts.xml` and `LocalClientFonts.xml`.

`ServerFonts.xml` and `ClientFonts.xml` contain standard WRS fonts. These files are located in the web application's WEB-INF directory. For example: C:\jboss-

4.2.0.GA\server\SASServer1\deploy_sas\sas.webreportstudio4.2.ear\sas.webreportstudio.war\WEB-INF.

`LocalServerFonts.xml` and `LocalClientFonts.xml` contain custom fonts. These files (if they exist) are located in customer subdirectory where WRS is installed. For example: C:\SAS\EBIServer\Lev1\Web\Applications\SASWebReportStudio4.2\customer.

Standard WRS Fonts

The `ServerFonts.xml` file lists fonts that are rendered on the server. These are the fonts that are available for graphs in a report. The fonts that are listed in this file should be installed on the middle-tier server where SAS® Web Report Studio is deployed.

The `ClientFonts.xml` file lists the fonts that are rendered on the client (user's) system. These fonts are available for tables, headers, and other text. These fonts should be installed on the machine where the browser is running.

The fonts in `ServerFonts.xml` and `ClientFonts.xml` follow this format:

```
<font actualfont = "Arial"    displayfont = "Arial"    displaykey =
"style.font.arial.txt” />
```

The attribute, `actualfont`, is the font name that is stored into the report. The attribute, `displayfont`, is the font name that shows in the font picker except for when it is overridden by `displaykey`. The attribute `displaykey` is a key into the `CitationWebMessages.properties` resource file where strings are stored so that they can be translated.

`CitationWebMessages.properties` is in `sas.webreportstudio.nls.jar` which locates in the web application’s WEB-INF\lib directory. For example: C:\jboss-4.2.0.GA\server\SASServer1\deploy_sas\sas.webreportstudio4.2.ear\sas.webreportstudio.war\WEB-INF\lib\sas.webreportstudio.nls.jar. The classpath is: com\sas\apps\citation\CitationWebMessages.properties

SAS provides default fonts in ServerFonts.xml for several locales. For example:

```
<font actualfont  = "Monotype Sans WT SC"
        locale      = "zh,zh_CN"
        displayfont = "SAS Monospace"
        displaykey  = "style.font.sas.monospace.txt" />
```

The displayfont `SAS Monospace` will only be displayed in font picker when the browser’s language is zh_CN or zh. If there is no locale for the font definition segment, the displayfont will only be displayed for en_US.

Customers can define other fonts in LocalServerFonts.xml if the SAS fonts do not meet their requirements.

The fonts in `ServerFonts.xml` and `ClientFonts.xml` are not intended to be edited; however there may be situations where individual fonts may be deleted. The *SAS® 9.2 Intelligence Platform Web Application Administration Guide* documents how SAS Web Report Studio obtains the default font. Please check out "Make Fonts Available to SAS Web Report Studio" on page 203 in particular.

Custom WRS Fonts

`LocalServerFonts.xml` contains the custom fonts used by the font pickers on the graph properties dialog. The fonts in `LocalServerFonts.xml` should be installed on the mid-tier server.

`LocalClientFonts.xml` contains the custom fonts used by the font pickers on the other non graph properties dialogs (table properties for example). The fonts in `LocalClientFonts.xml` should be installed on the machine where the browser is running.
.

You can create these files from the `LocalServerFonts.xml.sample` and `LocalClientFonts.xml.sample` files that reside in the SAS-configuration-directory `\Lev1\Web\Applications\SASWebReportStudio4.2\customer` folder. To create the files:

- Open `LocalServerFonts.xml.sample` and save it using the name `LocalServerFonts.xml`.

- Open `LocalClientFonts.xml.sample` and save it using the name `LocalClientFonts.xml`.

The fonts in these files follow this format:

LocalClientFonts.xml

```
<font actualfont = "fontname"   displayfont = "fontname" />
```

LocalServerFonts.xml

```
<font actualfont = "fontname"   locale = "ll_RR"
displayfont = "fontname" />
```

The attribute, `actualfont`, is the font name that is stored into the report. The value for this attribute should match the name of the font on the system. If they differ, a font substitution can occur.
The attribute, `displayfont`, is the font name that shows in the font picker.

Note that this format does not have a `displaykey` attribute. Therefore the customer added font names cannot refer to strings in the `CitationWebMessages.properties` file (which can't be modified by customers) and therefore are not translated.

`Styles.xml` and `styles_<locale>.xml` files exist in `sas.report.jar`; SAS® Web Report Studio uses the settings in these files to control the locale sensitive font settings for TitleText, LabelText, ValueText etc. The fonts in `Styles.xml` are also displayed by the font pickers on the graph properties dialog.

SAS® Web Report Studio must be reconfigured and redeployed after the custom font files are created or modified.

It looks at fonts in this order: Customized configuration file ► standard configuration file ► style_<locale>.xml. The first occurrence of a font overrides any later references. This means, for example, if `LocalServerFonts.xml` has

```
<font actualfont  = "Monotype Sans WT SC"
      locale      = "zh_CN"
      displayfont = "SAS Monospace"  />
```
and

`ServerFonts.xml` has

```
<font actualfont  = "Monotype Sans WT SC"
      locale      = "zh_CN"
      displayfont = "Monotype"      />
```

`SAS Monospace` rather than `Monotype` will be displayed in the font picker.

Similarly, if more than one font has the same displayfont, the first read is the one that will be used; other font elements with the same displayfont will be silently ignored. This means, for example, if `LocalServerFonts.xml` has

```
<font actualfont  = "Monotype Sans WT SC"
      locale = "zh,zh_CN"
      displayfont = "SAS Monospace" />
```
and

`ServerFonts.xml` has

```
<font actualfont  = "SAS Monospace "
      locale = "zh,zh_CN"
      displayfont = "SAS Monospace" />
```

`Monotype Sans WT SC` will be used for `SAS Monospace` since that is the first definition encountered.

In the UNIX environment, fonts must be installed correctly and loaded by the JVM in order for SAS® Web Report Studio to render them correctly. During the installation and configuration of SAS 9.2, some of the TrueType fonts are installed to the required Java Runtime Environment. When SAS 9.2 is configured to use an alternate JRE, these TrueType fonts must be installed to the new JRE. The default location of the JRE with UNIX installations is under the $SASHOME directory. The SASHOME directory is the root directory where the SAS 9.2 installation was performed. If you did not install the SAS JRE, you need to copy or move all of the SAS TrueType fonts from the fonts directory, which is located at {$ SASHOME }/SASFoundation/9.2/misc/fonts, to the fonts directory lib/fonts of the new JRE.

Please note that in all three types of the XML files above, according to the XML standard, special characters like ampersand (“&”) need to be escaped, that is, converted to character entities. The table below lists the characters together with their escape sequences.

Character	**Name**	**Escape Sequence**
&	Ampersand	&amp;
<	Left Angle Bracket	&lt;
>	Right Angle Bracket	&gt;
“	Quote	&quot;
‘	Apostrophe	&apos;

Fonts in PDF generation

To ensure that special fonts in different languages display correctly in the PDF reports, follow these steps:

1. In SAS Management Console on the Plug-ins tab, navigate to Application Management ► Configuration Manager ► Web Report Studio 4.2.
2. Right-click and select Properties to display the Web Report Studio 4.2 Properties dialog box.
3. Click on the Advanced tab.
4. Click Add to display the Define New Property dialog box.
5. Enter the property name as shown (note that there is a period at the beginning of the property name), along with the property value as shown:
   Property Name:
   `.vmwide.com.sas.report.render.view.pdf.itext.font.fontDirectories`
   Property Value: `Font_path_string`
   To specify multiple font directory paths in the UNIX environment, follow this format: Font_path_string1:Font_path_string2:Font_path_string3. In the Windows environment, follow this format:
   Font_path_string1;Font_path_string2;Font_path_string3
6. Click OK to exit the Define New Property dialog box.
7. Click OK to exit the Web Report Studio 4.2 Properties dialog box.
8. To enable this property to go into effect, restart your Web application server.

The fonts are searched in the order the directories that are specified below:

- c:/windows/fonts
- c:/winnt/fonts
- d:/windows/fonts
- d:/winnt/fonts
- /usr/X/lib/X11/fonts/TrueType
- /usr/openwin/lib/X11/fonts/TrueType
- /usr/share/fonts/default/TrueType
- /usr/X11R6/lib/X11/fonts/ttf

If the font used is not in the above default search path, you need to use the SAS Management Console and add the property as shown above in the Define New Property dialog box.

Property Name:
`.vmwide.com.sas.report.render.view.pdf.itext.font.fontDirectories`
Property Value: `Font_path_string`

To enable bi-directional printing add the following property in the Define New Property dialog box:
Property Name:
`.vmwide. com.sas.report.render.view.pdf.itext.BidiMode`
Property Value: `TRUE`

SAS/CONNECT® server configuration

With the 9.2 SAS/CONNECT® server, you can easily transfer data between SAS client(s) and server(s). It works well with multilingual data under a Unicode environment. If you do not run in a Unicode environment, you need to use the same (or a compatible) encoding on either side to prevent encoding incompatibilities.

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

An ordered table that shows the characters within a character set, along with their unique numeric representations. A code page captures an encoding, which results from applying an encoding method to a character set. For example, the German EBCDIC code page contains the encoding that results from applying the EBCDIC encoding method to the German character set.

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 Latin1 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.

font

A complete set of all the characters of the same design and style. The characters in a font can be figures or symbols as well as alphanumeric characters. There is a 1:n relationship between a character and its representation in different fonts.

legacy encoding

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.

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.

localization

The process of adapting a product to meet the language, cultural, and other requirements of a specific target environment or market so that customers can use their own languages and conventions when using the product. Translation of the user interface, system messages, and documentation is a large part of the localization process.

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 Latin1 encoding under UNIX to a German EBCDIC encoding on an IBM mainframe.

Unicode

An encoding standard that supports the interchange, processing, and display of characters and symbols used in writing all of the world's languages. The Unicode Standard defines three Unicode encoding forms: UTF-8, UTF-16, and UTF-32.

UTF-8

A varying-length form of Unicode where ASCII characters are all represented in 1 byte. Other characters take from two to four bytes to represent.

---

[1] New terms (when first appearing) in the paper are explained in the Glossary.
[2] SAS windowing environment is an interactive windowing interface to SAS software. In this environment you can issue commands by typing them on the command line, by pressing function keys, or by selecting items from menus or menu bars. It used to be called the SAS Display Manager System.
[3] All names in this paper are purely fictitious. Any resemblance to actual persons is completely accidental.
[4] Be careful. If a data set already exists SAS uses the encoding stored in the original data set!
[5] The terms character set, code page, and encoding are often used interchangeably or as synonyms: This is not quite correct but for sake of simplicity, we assume that they are synonyms.
[6] CL: Column Length defined in Oracle database; VL: Variable Length defined in a SAS data set; National_max_bytes: maximal bytes per character value of the database national character set.

[7] BiDi (bidirectional) text is a mixture of Arabic or Hebrew text with Latin text or numbers. Arabic and Hebrew strings of text, for example, are read from right to left, but numbers and embedded Latin text strings are read from left to right.