Migrating Data to UTF-8 for the SAS® Viya® Platform

2020.1 - 2024.06

This document might apply to additional versions of the software. Open this document in SAS Help Center and click on the version in the banner to see all available versions.

UTF-8 Encoding

UTF-8 is part of the Unicode coded character set. UTF-8 is becoming the preferred and most-used encoding, and it is the recommended encoding for using Unicode with operating systems like Linux.

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

UTF-8 runs on SAS under Windows and UNIX, but not z/OS.

Table 1 *Width Character Set*

Bytes	Characters
1	US_ASCII Characters
2	East and West European, Baltic, Greek, Turkish, Cyrillic, Hebrew, Arabic
3	Chinese, Japanese, Korean (CJK), Thai, Indic, and certain control characters
4	Uncommon CJK characters and various historic scripts

You should be aware of the encoding of your data sets. You might have data sets that are not UTF-8 encoded. Determine the encoding of your data and then migrate the data to UTF-8 if needed.

Migrating Data to UTF-8 Encoding

You can migrate your data from SAS using other encodings to SAS using UTF-8 encoding in order to support multilingual data and to support SAS Viya.

You can use these tasks to migrate data to UTF-8:

1 “Determine the Encoding of a Data Set”.
2 “Migrating Data from WLATIN1 to UTF-8”.
3 “Determine Storage Size Requirements”.
4 “Use CEDA to Read Data”.
5 “Determine Whether the CVP Engine Is Needed to Read Your Data without Truncation”.
6 “Migrating Data to UTF-8”.

7 “Convert Format Catalogs to UTF-8”.

8 “Read External Files”.

Determine the Encoding of a Data Set

You can use PROC CONTENTS to determine the encoding of your data set. The following code displays information about the data set header:

```
proc contents data=sashelp.class;
run;
```

Here is the output from the PROC CONTENTS code. The encoding is US-ASCII.

```
The CONTENTS Procedure

      Data Set Name        SASHELP.CLASS                                  Observations          19
      Member Type          DATA                                           Variables             5
      Engine               V9                                             Indexes               0
      Created              01/17/2016 20:12:45                            Observation Length    40
      Last Modified        01/17/2016 20:12:45                            Deleted Observations  0
      Protection                                                          Compressed            NO
      Data Set Type                                                       Sorted                NO
      Label                Student Data
      Data Representation  SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64
      Encoding             us-ascii  ASCII (ANSI)
```

Migrating Data from WLATIN1 to UTF-8

Migrating data from WLATIN1 to UTF-8 is a very common data migration. Here are two simple ways to migrate data from WLATIN1 to UTF-8. For more information, see “Migrating Data from WLATIN1 to UTF-8” in *SAS National Language Support (NLS): Reference Guide*.

Here are examples of migrating a single data set and an entire library of data sets from WLATIN1 to UTF-8.

Run the following code from a UTF-8 SAS session.

```
libname wlt1 CVP ".\wlatin1";    1
libname u8 ".\utf8";    2
```

1 Create a libref for the library containing the WLATIN1 data set. Access the CVP engine to expand character variables to avoid truncation.

2 Create a libref for the output library.

The following code migrates a single data set.

```
data u8.latin_names;    1
```

```
   set wlt1.latin_names;
run;
```

1 Read and copy the data set from WLATIN1 to UTF-8.

When migrating an entire library you need the SAS session encoding value. The encoding for the SAS session value is shown by the PROC OPTIONS code. If the library contains data sets that might have a different encoding, then the outencoding= option is necessary in the LIBNAME statement for U8. The encoding value is used in the U8 library provided the library is empty.

```
PROC OPTIONS OPTION=ENCODING
run;
```

Ensure that OUT is empty.

```
proc datasets lib=out kill;
run;
quit;
```

The following code migrates an entire library of data sets.

```
proc datasets library=wlt1;    1
   copy out=u8 NOCLONE;
run;
```

1 Migrate all data sets in wlt1 to U8 using PROC DATASETS.

Determine Storage Size Requirements

Storing text as a UTF-8 encoding might take more space than storing it in legacy encodings. The expansion amount depends on the language and text. Here are some possible expansions for some common legacy encodings:

Table 2 *Examples of Storage Size Increases in UTF-8*

Encoding	Languages	Storage Size Increase in UTF-8
ASCII	English, Malay	0%
ISO-8859-1	Western European	10%
ISO-8859-7, plain text	Greek	90%
ISO-8859-7, 50% markup	Greek	45%
TIS-620, plain text	Thai	190%
TIS-620, 50% markup	Thai	95%
EUC-KR, plain text	Korean	50%

Encoding	Languages	Storage Size Increase in UTF-8
EUC-KR, 50% markup	Korean	55%

Footnote: Adapted from W3C Internationalization (I18n) Activity.

Use CEDA to Read Data

Cross-environment data access (CEDA) enables a SAS file that was created in a directory-based operating environment (for example, Linux) to be processed in a different operating system, a different session encoding, or both. With CEDA, the processing is automatic and transparent. In many cases, you do not need to create a transport file, use SAS procedures that convert the file, or change your SAS program.

When the source and session encodings are not the same, CEDA reads and converts the data. When the UTF-8 session reads a data set created with a LATIN1 encoding, you see the following note in your log:

```
NOTE: Data file MYLIB.CARS.DATA is in a format that is native to another host,
 or the file encoding does not match the session encoding.
 Cross  Environment Data Access will be used, which might require additional
 CPU resources and might reduce performance.
```

If all of the characters in the data sets are ASCII characters, using the data should be easy because the first 128 characters of Unicode correspond one-to-one with ASCII. Performance might be affected, however, because the data set encoding does not match the session encoding and CEDA must transcode every record.

You do not have to use CEDA if your data set exclusively contains English text that is ASCII only: the English alphabet, digits from 0 to 9, and punctuation characters. You can change the encoding of your data set to ASCIIANY to prevent transcoding. Use the CORRECTENCODING option in the PROC DATASETS MODIFY statement to change the encoding that is stored in the data set header. Here is an example of specifying ASCIIANY:

CAUTION
Misuse of ASCIIANY could result in data corruption. ASCIIANY prevents transcoding. Specify this option only if your data exclusively contains ASCII data.

```
libname myfiles "path to data sets";
proc datasets library=myfiles;
   modify olddata / correctencoding=ASCIIANY;
quit;
```

Note: Transcoding can result in character-data loss when encodings are incompatible. For information about encoding and transcoding, see *SAS National Language Support (NLS): Reference Guide*.

Note: If the data set contains non-ASCII characters, extended ASCII characters, or characters 128-255 in single-byte character sets, you must continue using CEDA or convert the data set to UTF-8. You might need to use the CVP LIBNAME engine to enlarge the variable column width. These characters need 2 to 4 bytes in UTF-8 encoding. See "Determine Whether the CVP Engine Is Needed to Read Your Data without Truncation" on page 6 for more information about the CVP LIBNAME engine.

Note: If you used a Microsoft product to create your data, quotation marks and hyphens might have been converted to smart quotation marks or hyphens. These characters are not ASCII characters. A transcoding error might appear. These characters require more than 1 byte when they are converted from Latin1 to UTF-8.

Note: Indexes on the data set are not used when reading the data. See "Migrating Data to UTF-8" on page 8.

Determine Whether the CVP Engine Is Needed to Read Your Data without Truncation

When you read or write a data set that has a different encoding, you might see a warning or an error like this:

```
WARNING: Some character data in the data set "XXXXXXX" was lost
 during transcoding. Truncation occurred during transcoding to the
new encoding. To avoid the transcoding error, please refer to
"Troubleshooting Truncation and Data loss Issue during
Transcoding" in SAS National Language Support (NLS): reference guide.
```

This message usually means that there is not enough space in one or more character columns in the observation buffer of the data set to convert the data to UTF-8.

Truncation can occur when characters in the original encoding are converted to an encoding that requires more bytes to represent those same characters. For example, when characters that are encoded as LATIN1, where every character is represented using 1 byte, are transcoded to UTF-8, where some of the characters require 1, 2, or even 3 bytes, truncation can occur if the character column is not wide enough.

To solve the truncation problem, use the character variable padding (CVP) engine with the LIBNAME statement to read the data set. The CVP engine adds padding to the character columns. By default, the character variable lengths are multiplied by 1.5. To specify a different expansion amount, use the CVPMULTIPLIER= option. You can specify a multiplier value from 1 to 5, or you can specify a value of 0 and then let the CVP engine determine the multiplier.

Note: See the "CASNCHARMULTIPLIER= System Option" in *SAS System Options: Reference* for information about transcoding to UTF-8 in the CAS server.

Here is an example that uses the CVP engine with the default multiplier:

```
libname mylib CVP "path to data sets";
```

Here is an example using a different multiplier:

```
libname mylib CVP "path to data sets" cvpmultiplier=2.0;
```

Libraries accessed with the CVP engine are read-only. If you want to save a permanent copy of the data, you need to create a new data set. Use the following steps in a SAS session with UTF-8 encoding to create a new data set:

1 The first LIBNAME statement points to the original data set. Use a second LIBNAME statement to point to the location of the library that will contain the new data set.

```
libname mylib cvp "path to original data set";
libname mylib2 "path to new data set";
```

2 Use PROC DATASETS with the COPY statement and the OVERRIDE= option. When you specify `OVERRIDE=(ENCODING=SESSION OUTREP=SESSION)` in the COPY statement, the new data set is created in the host data representation and encoding of the SAS session that is executing the COPY statement. Add the CONTENTS statement to view a description of the content of the new data set.

```
proc datasets nolist;
  copy in=mylib out=mylib2 override=(encoding=session outrep=session);
  contents data=mylib2.mydata;
run;
```

Here is a portion of the output when the example code is run on Linux using UTF-8 encoding:

Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64
Encoding	utf-8 Unicode (UTF-8)

If formats are attached to your data variables, you can extend the format width and variable length to avoid truncation. You can also disable the extension with `CVPFORMATWIDTH=NO`. For more information, see "Convert Format Catalogs to UTF-8" on page 8.

For more information see "Avoiding Character Data Truncation By Using the CVP Engine" in *SAS National Language Support (NLS): Reference Guide*, "COPY Procedure" in *Base SAS Procedures Guide* and "LIBNAME Statement: CVP Engine" in *SAS Global Statements: Reference*.

As an alternative to using the CVP engine, you can use the %COPY_TO_UTF8 macro to avoid character truncation. For more information see "%COPY_TO_UTF8 Macro" in *SAS National Language Support (NLS): Reference Guide*

Use the `CONSTRAINT=YES` value if some data sets contain integrity constraints. The `MT=DATA` value restricts the copy to SAS files of type data. If you have views, they can be copied with `MT=VIEW`, but DATA step views are not recompiled for the new host until they are referenced. They must contain their source to be recompiled.

Larger buffer sizes that are made to match host logical volume transfer sizes to storage and physical storage block sizes affect SAS IO throughput rates. In SAS, the default BUFSIZE option is 64K, and it works well with most storage configured for large block operations. The transfer size can be storage-dependent. Consult your site administrator for the storage configuration. There are a few storage platforms that optimize on 128K, some at 256K, and some at 1 MB. Flash arrays can present differently than HDD arrays.

Consult with your host and storage administration staff to ensure that you are using maximal host-to-storage bandwidth and that you are using bandwidth between old and new systems. Ensure that your backup and redundancy plan is in place and working properly (for example, run recovery tests on sample data).

Migrating Data to UTF-8

Indexes and integrity constraints that are created with an encoding that differs from the session encoding cannot be used. However, you can capture the definitions of the indexes and integrity constraints in the original data set and re-create them in a new SAS session with UTF-8 encoding.

PROC MIGRATE can convert indexes to the new encoding as long as there is enough space in the character columns to convert the data to UTF-8.

1 Assign the source library by using CVP engine

```
libname inlib cvp 'sas9_dir';
```

2 Assign the target library

```
libname outlib 'viya_dir'
```

3 PROC MIGRATE migrates all data from the source library to the target library.

```
proc migrate in=inlib out=outlib;
run;
```

Note: If a transcoding error occurs when you run the program, increase the multiplier used by the CVP engine.

Convert Format Catalogs to UTF-8

SAS catalog entries that contain formats must be converted to UTF-8 encoding when truncation might occur because characters in the original encoding are converted to an encoding that requires more bytes to represent them. If a width is stored as part of the associated format, the width is not expanded by the CVP engine. The format can cause truncation when the formatted value is displayed.

Formats are stored in the FORMATS catalog using the session encoding in which they were created. The FORMATS catalog in the following example is named formats. SAS assumes the extension of .sas7bcat. The full name of the catalog in the directory is formats.sas7bcat. The FORMATS catalog does not contain encoding information. The encoding information is in the data set (fmtloc.outfmts).

1 If the source libraray contains format cataglogs, specify the encoding of the cataglog files. PROC MIGRATE migrates the data.

```
proc migrate in=inlib out=outlib encoding='latin1';
run;
```

Note: For more information about user-defined formats with SAS Cloud Analytic Services (CAS), see "About User-Defined Formats" in *SAS Cloud Analytic Services: User-Defined Formats*.

The default value for the format width in the CNTLOUT data set is the default width (in bytes) for the format. The default value is computed based on the largest label unless you specify the DEFAULT=, MIN=, and MAX= options when you create the format. If the START, MIN, MAX, or LABEL variable (in characters) is larger in UTF-8 encoding, these widths are not expanded by the CVP engine. Use the %COPY_TO_ENCODING macro instead. For more information, see "%COPY_TO_NEW_ENCODING Macro" in *SAS National Language Support (NLS): Reference Guide*.

Read External Files

SAS reads and writes external files using the current session encoding and assumes that the external file uses the same encoding as the SAS session. When a file contains character data and its encoding is different from the SAS session encoding, use the ENCODING= option in the FILENAME, INFILE, or FILE statement to specify the file encoding. SAS can transcode the data from its original encoding to the current SAS session encoding. Character columns must be long enough to hold the data in the SAS session encoding. Otherwise, truncation can occur.

Here is an example:

```
FILENAME myfn "path and file name" ENCODING=LATIN1;
```

You can run PROC CIMPORT in your SAS UTF-8 session to import data sets that were created using any encoding. If the transport file contains a data set that is not encoded as UTF-8 or US-ASCII, the SAS UTF-8 session transcodes the data to UTF-8. If the columns in the data set are not long enough to hold the data that was transcoded to UTF-8, a warning is written to your SAS log stating that the destination buffer size is not sufficient for the transcoded data. SAS continues to read the data set. However, the character strings are truncated to fit within the number of bytes available in the character column.

To ensure that your destination buffer size is sufficient for the transcoded data, you can specify the EXTENDVAR= and the EXTENDFORMAT = options on PROC CIMPORT. For more information, see "CIMPORT Procedure" in *Base SAS Procedures Guide*.