# Contents


## PART 1  NLS Concepts  1

**Chapter 1 / National Language Support (NLS)** ........................................... 3  
Overview to National Language Support .................................................. 3  
Definition of Localization and Internationalization .................................. 4

**Chapter 2 / Locale for NLS** ................................................................. 5  
Overview of Locale Concepts for NLS .......................................................... 5  
Specifying a Locale ..................................................................................... 6

**Chapter 3 / Encoding for NLS** .............................................................. 9  
Overview: Encoding for NLS ...................................................................... 9  
Difference between Encoding and Transcoding ......................................... 10  
Common Encoding Methods ....................................................................... 10  
Standards Organizations for NLS Encodings ............................................. 14  
Code Point Discrepancies among EBCDIC Encodings ................................ 14  
Collating Sequence ................................................................................... 15  
Determining the Encoding of a SAS Data Set .......................................... 19  
Default SAS Session Encoding .................................................................. 20  
Setting the Encoding of a SAS Session ..................................................... 21  
Encoding Behavior in a SAS Session .......................................................... 23  
Migrating Data to UTF-8 ........................................................................... 24  
Migrating Data from WLATIN1 to UTF-8 .................................................. 25

**Chapter 4 / Transcoding for NLS** .......................................................... 31  
Overview of Transcoding ......................................................................... 31  
Common Reasons for Transcoding ............................................................. 32  
Transcoding and Translation Tables ............................................................. 32  
SAS Options That Transcode SAS Data ....................................................... 34  
Transcoding Considerations ...................................................................... 34  
Compatible and Incompatible Encodings .................................................... 35  
Avoiding Character Data Truncation By Using the CVP Engine .................. 41  
Avoiding Character Truncation Using the %COPY_TO_NEW_ENCODING Macro 42

**Chapter 5 / Double-Byte Character Sets (DBCS)** .................................. 45  
Overview to Double-Byte Character Sets (DBCS) ..................................... 45  
East Asian Languages ............................................................................... 46  
Specifying DBCS ...................................................................................... 46  
Requirements for Displaying DBCS Character Sets .................................... 46  
When You Can Use DBCS Features ............................................................ 47  
DBCS and SAS on a Mainframe ................................................................. 47
<table>
<thead>
<tr>
<th>Chapter 6 / Time Zones</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of SAS Time Zones</td>
<td>51</td>
</tr>
<tr>
<td>Specifying Time Zones in SAS</td>
<td>51</td>
</tr>
<tr>
<td>PART 2 Autocall Macros for NLS</td>
<td>61</td>
</tr>
<tr>
<td>Chapter 7 / Dictionary of Autocall Macros for NLS</td>
<td>63</td>
</tr>
<tr>
<td>Autocall Macro Entries by Category</td>
<td>63</td>
</tr>
<tr>
<td>Dictionary</td>
<td>64</td>
</tr>
<tr>
<td>PART 3 Data Set Options for NLS</td>
<td>67</td>
</tr>
<tr>
<td>Chapter 8 / Dictionary of Data Set Options for NLS</td>
<td>69</td>
</tr>
<tr>
<td>Data Set Options by Category</td>
<td>69</td>
</tr>
<tr>
<td>Dictionary</td>
<td>69</td>
</tr>
<tr>
<td>PART 4 Formats for NLS</td>
<td>75</td>
</tr>
<tr>
<td>Chapter 9 / Overview to NLS Formats</td>
<td>77</td>
</tr>
<tr>
<td>International Date and Datetime Formats</td>
<td>77</td>
</tr>
<tr>
<td>Currency Representation</td>
<td>78</td>
</tr>
<tr>
<td>Exceptions for Date and Time Default Widths</td>
<td>85</td>
</tr>
<tr>
<td>Chapter 10 / Dictionary of Formats for NLS</td>
<td>89</td>
</tr>
<tr>
<td>Categories of NLS Formats</td>
<td>93</td>
</tr>
<tr>
<td>Dictionary</td>
<td>116</td>
</tr>
<tr>
<td>PART 5 Functions for NLS</td>
<td>341</td>
</tr>
<tr>
<td>Chapter 11 / Internationalization Compatibility for SAS String Functions</td>
<td>343</td>
</tr>
<tr>
<td>Internationalization Compatibility for SAS String Functions</td>
<td>343</td>
</tr>
<tr>
<td>Chapter 12 / VARCHAR Support in Functions</td>
<td>371</td>
</tr>
<tr>
<td>VARCHAR Data Type in String Functions</td>
<td>371</td>
</tr>
<tr>
<td>Chapter 13 / Dictionary of Functions for NLS</td>
<td>375</td>
</tr>
<tr>
<td>Functions by Category</td>
<td>376</td>
</tr>
<tr>
<td>Dictionary</td>
<td>382</td>
</tr>
</tbody>
</table>
## PART 6  Informats for NLS  517

**Chapter 14 / Dictionary of Informats for NLS**
- Informats by Category  521
- Dictionary  535

## PART 7  Macro Functions for NLS  667

**Chapter 15 / Dictionary of Macro Functions for NLS**
- Macro Functions by Category  669
- Dictionary  670

## PART 8  System Options for NLS  681

**Chapter 16 / Dictionary of System Options for NLS**
- System Option Entries by Category  683
- Dictionary  685

## PART 9  Options for Commands, Statements, and Procedures for NLS  719

**Chapter 17 / Dictionary of Command, Statement, and Procedure Option for NLS**
- Commands, Statements, and Procedures for NLS by Category  721
- Dictionary  722

## PART 10  Procedures for NLS  759

**Chapter 18 / DBCSTAB Procedure**
- Overview: DBCSTAB Procedure  761
- Syntax: DBCSTAB Procedure  761
- Examples: DBCSTAB Procedure  763

**Chapter 19 / LOCALEDATA Procedure**
- Overview: LOCALEDATA Procedure  767
- Syntax: LOCALEDATA Procedure  767
- Examples: LOCALEDATA Procedure  776

**Chapter 20 / TRANTAB Procedure**
- Overview: TRANTAB Procedure  783
- Concepts: TRANTAB Procedure  784
- Syntax: TRANTAB Procedure  787
**PART 11** Values for Locale, Encoding, and Transcoding 809

**Chapter 21 / Values for the LOCALE= System Option** 811
- LOCALE= Values for PAPERSIZE and DFLANG, Options 811
- Locale-to-Encoding Mapping 819

**Chapter 22 / SAS System Options for Processing DBCS Data** 827
- Overview to System Options Used in a SAS Session for DBCS 827
- DBCS Values for a SAS Session 827

**Chapter 23 / Encoding Values in SAS Language Elements** 829
- Overview to SAS Language Elements That Use Encoding Values 829
- SBCS, DBCS, and Unicode Encoding Values for Transcoding Data 829

**Chapter 24 / Encoding Values for a SAS Session** 841
- UNIX Encoding Values 841
- Windows Encoding Values 842
- z/OS Encoding Values 844

**PART 12** Appendixes 849

**Appendix 1 / Additional NLS Language Elements** 851
- Additional NLS Language Elements 852

**Appendix 2 / Encoding and Alias Names** 943
- Encodings and Their Aliases and Encoding Character Set Compatibility 943

**Appendix 3 / Time Zone IDs and Time Zone Names** 953
- Area: Africa 953
- Area: America–North, Central, and South 956
- Area: Antarctica 966
- Area: Arctic 966
- Area: Asia 967
- Area: Atlantic 972
- Area: Australia 973
- Area: Miscellaneous 975
- Area: Europe 981
- Area: Pacific 988

**Appendix 4 / %COPY_TO_NEW_ENCODING Macro** 991
What's New in SAS 9.4 National Language Support

Overview

This release expands the scope and capabilities of SAS National Language Support (NLS). NLS is a set of features that enables a software product to function properly in every global market for which the product is targeted. SAS contains NLS features to ensure that you can write SAS applications that conform to local language conventions. Typically, software that is written in the English language works well for users who speak English and for data that is formatted using the conventions that are observed in the United States. However, without NLS, these products might not work as well for users in other regions of the world. SAS NLS enables users in regions such as Asia and Europe to process data successfully in their native languages and environments.

General Enhancements

The following enhancements are implemented for SAS 9.4 National Language Support:

- “Exceptions for Date and Time Default Widths” on page 85 provides exceptions to default widths. Some format widths exceed the default width, depending on the locale and encoding.

- The SAS name and POSIX name, Serbian_Yugoslavia sr_YU, has been removed from the document.

- “Specifying Time Zones in SAS” on page 51 explains SAS time zones.

- Appendix 3, “Time Zone IDs and Time Zone Names,” on page 953 lists time zone IDs and time zone names.

SAS 9.4M2 has the following new chapter. Time zone information was taken from the overview and added to this chapter.

- Chapter 6, “Time Zones,” on page 51

SAS 9.4M4, has the following changes:

- Chapter 4, “Transcoding for NLS” is enhanced to reflect compatible encodings and troubleshooting tips to process encodings that are not compatible.
A new appendix, "Encodings and Their Aliases and Encoding Character Set Compatibility" on page 943 is added.

In SAS 9.4M5, information about avoiding character truncation using a SAS macro has been added. See "Avoiding Character Truncation Using the %COPY_TO_NEW_ENCODING Macro" on page 42 and "%COPY_TO_NEW_ENCODING Macro Function" on page 991.

In SAS 9.4M5, with the addition of "%COPY_TO_NEW_ENCODING Macro Function" on page 991, the restriction that the encoding of the data set must be the same as the session encoding is lifted.

## Additional Encodings

The following encodings are new:

- LATIN7
  - Baltic Rim
- LATIN10
  - South-Eastern European

## Data Set Options

In 9.4M7, the SESSION option is added to the "ENCODING= Data Set Option".

## Locales

The following locales were added:

- English_Malta (en_MT)
- Greek_Cyprus (el_CY)
- Irish_Ireland (ga_IE)

For more information, see the LOCALE= table on page 811.

In SAS 9.4M3 the Manx_Gaelic_UnitedKingdom (gv_GB) locale was deprecated.

SAS 9.4M3 has the following new locales:

- Kazakh_Kazakhstan (kk_KZ)
- Basque_Spain (eu_ES)

In SAS 9.4M5, the Tagalog-Philippines locale was added.

In SAS Viya 3.5, aliases and POSIX numbers were updated for these locales. See the LOCALE= table on page 811.
 Formats

The following formats are new:

**B8601DX**
Converts UTC datetime values into user local time and writes local time (SAS datetime values) with time zone offsets.

**B8601LX**
Writes SAS datetime values with time zone offsets.

**B8601TX**
Converts UTC time into user local time and writes local time values with time zone offsets.

**E8601DX**
Converts UTC datetime values into user local time and writes local time (SAS datetime values) with time zone offsets.

**E8601LX**
Writes SAS datetime values with time zone offsets.

**E8601TX**
Converts UTC time into user local time and writes local time values with time zone offsets.

**NLDATEL**
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date in the form month, date, year.

**NLDATEM**
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date.

**NLDATEMDL**
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.

**NLDATEMDM**
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.

**NLDATEMDS**
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.
NLDATES
   Converts a SAS date value to the date string of the specified locale and then writes the date value as a date string.

NLDATEYML
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and year.

NLDATEYMM
   Converts a SAS date value to the date string of the specified locale and then writes the date values as the month and year with abbreviations.

NLDATEYMS
   Converts a SAS date value to the date string of the specified locale and then writes the date value as a date and year.

NLDATEYQL
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.

NLDATEYQM
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.

NLDATEYQS
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value (1–4) with numbers and delimiters.

NLDATML
   Converts a SAS date value to the date string of the specified locale and then writes the date value as a date in the form month, date, year, and time.

NLDATMM
   Converts a SAS date value to the date string of the specified locale and then writes the date value as a date and time with abbreviations for the month and time.

NLDATMMDL
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.

NLDATMYQS
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value using numbers and delimiters.

NLDATMMDM
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month using abbreviations.

NLDATMMDS
   Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month using numbers and delimiters.

NLDATMS
   Converts a SAS date value to the date string of the specified locale and then writes the date value as a date in the form MM/DD/YYYY.
NLDATMYML
Converts a SAS date value to the date string of the specified locale and then Writes the date value as the month and the year.

NLDATMYMM
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and the year.

NLDATMYQL
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year’s quarter value (1–4) and the year.

NLDATMYQM
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year’s quarter value (1–4) and the year.

NLDATMYQS
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the quarter (1-4) using numbers and delimiters.

NLDATMYMS
Converts a SAS date value to the date string of the specified locale and then writes the month and year with numbers and delimiters.

SAS 9.4M2 has the following new format:

BESTDOTX
Specifies that SAS choose the best notation and use a dot as a decimal separator.

In SAS 9.4M2, the country Latvia was added to the following formats:

NLMNIEUR
Writes the monetary format of the international expression for Belgium, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

NLMNLEUR
Writes the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

In SAS 9.4M3 the following formats were modified:

NLDATEYM
Includes a note and example explaining how to use PROC LOCALEDATA to process data with a width of 6.

NLMNIEUR
Writes the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

NLMNLEUR
Writes the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

In SAS 9.4M5, the following format is new:

DTWEEKV
Writes a week and datetime number in decimal format by using the V algorithm.
In SAS Viya 3.5, these formats are new:

**NLTIMEL**
Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value in a long-uniform pattern.

**NLTIMEM**
Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value in a medium-uniform pattern.

**NLTIMES**
Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value in a short-uniform pattern.

In SAS Viya 3.5, documentation for these formats are added:

**JNENGO**
Writes SAS date values as Japanese dates as reign, year, month, and day.

**JNENGOT**
Writes SAS datetime values as Japanese datetimes as reign, year, month, day, hour, and minute, and with narrow characters for numbers.

**JNENGOTW**
Writes SAS datetime values as Japanese datetimes as reign, year, month, day, hour, and minute, and with wide characters for numbers.

**JNENGOW**
Writes SAS date values as Japanese dates as reign, year, month, and day, and with wide character for numbers.

In SAS 9.4M7, this format is updated:

**YEN**
a restriction stating that the YEN format does not support UTF8 encoding is added.

---

**Functions**

The following functions are new:

**BASECHAR**
Converts characters to base characters.

**KSTRIP**
Removes leading and trailing blanks from a character string.

**SETLOCALE**
Specifies the locale keys for the current SAS locale.

**TZONEID**
Returns the current time zone ID.

**TZONENAME**
Returns the current standard (or daylight saving time) time zone name.

**TZONES2U**
Converts a SAS datetime value to a UTC datetime value.
TZONEU2S
Converts a UTC datetime value to a SAS datetime value.

An example was added in the following functions:
- “KCOMPARE Function” on page 403
- “KCOMPRESS Function” on page 405
- “KCOUNT Function” on page 407
- “KINDEX Function” on page 425
- “KINDEXC Function” on page 428
- “KLEFT Function” on page 430
- “KLENGTH Function” on page 432
- “KLOWCASE Function” on page 433
- “KREVERSE Function” on page 443
- “KRIGHT Function” on page 444
- “KSCAN Function” on page 445
- “KSTRCAT Function” on page 451
- “KSUBSTR Function” on page 454
- “KSUBSTRB Function” on page 455
- “KTRANSLATE Function” on page 458
- “KTRIM Function” on page 460
- “KTRUNCATE Function” on page 462
- “KUPCASE Function” on page 463
- “KUPDATE Function” on page 464
- “KUPDATEB Function” on page 466
- “KVERIFY Function” on page 470

In SAS 9.4M1, the following function is new:
ANORM420
Returns a normalized string from an input string encoded in EBCDIC420.

SAS 9.4M2 has the following new functions:
KUPDATES
Inserts, deletes, and replaces character value contents.

TZONEDSTNAME
Returns a daylight saving time name.

TZONEDSTOFF
Returns the time zone offset value for the specified daylight saving time.

TZONESTTNAME
Returns a standard time zone name.

TZONESTTOFF
Returns the time zone offset value for the specified standard time.

In SAS 9.4M2, the following functions were updated:
KCVT
DBCS, SBCS, and MBCS information was added and references to DBCSLANG and DBCSTYPE were removed.

KUPDATE
The NLSCOMPATMODE option was removed from the function.

SAS 9.4M3 has the following new functions:

KINDEXB
Searches a character expression for a string of characters.

KINDEXCB
Searches a character expression for specified characters.

KVERIFYB
Returns the position of the first character that is unique to an expression.

In SAS 9.4M3 the I18N level changed for the following functions:

ANORM420
I18NL1

BASECHAR
I18NL2

GETLOCPENV
I18NL2

GETPXLANGUAGE
I18NL2

GETPXLOCALE
I18NL2

GETPXREGION
I18NL2

KCOMPARE
I18NL2

KCOMPRESS
I18NL2

KCOUNT
I18NL2

KCVT
I18NL1

KINDEX
I18NL2

KINDEXC
I18NL2

KLEFT
I18NL2

KLENGTH
I18NL2

KLOWCASE
I18NL2

KPROPCASE
I18NL2
KPROPCHAR  I18NL2
KPROPDATA  I18NL2
KREVERSE   I18NL2
KRIGHT     I18NL2
KSCAN      I18NL2
KSTRCAT    I18NL2
KSTRIP     I18NL2
KSUBSTR    I18NL2
KSUBSTRB   I18NL2
KTRANSLATE I18NL2
KTRIM      I18NL2
KTRUNCATE  I18NL2
KUPCASE    I18NL2
KUPDATE    I18NL2
KUPDATEB   I18NL2
KUPDATES   I18NL2
KVERIFY    I18NL2
NLDATE     I18NL2
NLDATEM    I18NL2
NLTIME     I18NL2
SASMSG     I18NL2
SASMSGL    I18NL2
In **SAS 9.4M3**, SAS supports the Unicode supplementary character sets with a Unicode session. The following functions were updated:

- “UNICODE Function” on page 507
- “UNICODEC Function” on page 509
- “UNICODEWIDTH Function” on page 511

In **SAS 9.4M4**, the short description and details section for the following functions changed *letter to single-width English alphabet*.

- “KLOWCASE Function” on page 433
- “KUPCASE Function” on page 463

**SAS 9.4M5** has the following new CALL routine:

**CALL KSCANX**

Returns the position and length of the *n*th word from a character string.
SAS 9.4M5 has the following new functions:

**KCHARLIST**
Returns the character list according to the modifier.

**KCOUNTC**
Counts individual characters in a character string.

**KCOUNTW**
Counts the number of words in a character string.

**KCOUNTX**
Counts the number of times that a specified substring appears within a character string.

**KFIND**
Searches for a specific substring of characters within a character string.

**KFINDC**
Searches a string for any character in a list of characters.

**KFINDW**
Returns the character position of a word in a string or the number of the word in a string.

**KSCANX**
Selects a specified word from a character expression.

**KSUBSTRN**
Returns a substring, allowing a result with a length of 0.

SAS 9.4M5 has the following updated functions:

**KLEFT**
The Unicode Spaces That Are Removed by KLEFT, KRIGHT, and KTRIM table is new.

**KSTRIP**
Information about data normalization has been added.

SAS 9.4M6 has the following updated function:

**KCOMPRESS**
The `modifier(s)` argument is added.

In SAS Viya 3.5, these functions were updated:

**KCOMPARE**
Modifiers were added.

**KPROPDATA**
The PUNC argument was added.

Informats

In SAS 9.4M2, the country Latvia was added to the following informats:
NLMNIEUR
Reads the monetary format of the international expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

NLMNLIEUR
Reads the monetary format of the local expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

SAS 9.4M2 has the following new informat:

NLDATEW
Reads the date value in the specified locale and then converts the date value to the local SAS date and the day of the week.

NLDATMW
Reads the date value in the specified locale and then converts the date value to the local SAS day of the week and the datetime.

NLDATMAP
Reads the date value in the specified locale and then converts the date value to the local SAS datetime with either a.m. or p.m.

In SAS 9.4M2 aliases were added to the following informat:

NLDATE
Reads the date value in the specified locale and then converts the date value to the local SAS date value.

NLTIME
Reads the time value in the specified locale and then converts the time value to the local SAS time value.

NLDATM
Reads the datetime value of the specified locale and then converts the datetime value to the local SAS datetime value.

In SAS 9.4M3, the following informat were updated:

NLMNIEUR
Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

NLMNLIEUR
Reads the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

SAS 9.4M5 has the following new informat:

NLSTRMON
Reads the month name in the specified locale and converts it to a numeric value.

In SAS Viya 3.5, documentation is added for this informat:

JNENGO
Reads Japanese kanji date values in the form yymmdd.

In SAS 9.4M7, this informat is updated:
YEN
  a restriction stating that the YEN informat does not support UTF8 encoding is added.

Macro Functions

In SAS 9.4M5, the following macro function is new:

VALIDCHS
  Validates the character(s) encoding compatibility for data set variables.

Procedures

The following procedure is new:

LOCALEDATA
  Enables you to customize locale data.

Statement Options

In SAS 9.4M5, the following option is new:

CVPFORMATWIDTH
  Extends the width of the format.

In SAS Viya 3.5, these options are new:

CVPEXCLUDE
  Specifies the variables that are excluded during data processing.

CVPINCLUDE
  Specifies the variables that are included during data processing.

System Options

The following system options are new:

LSWLANG
  Specifies the language for the language switching feature when the
  LOGLANGCHG or ODSLNGCHG system option is set at SAS invocation.
LOGLANGENG
  Specifies using the English language for SAS log message text when the
  LOCALE option is set after start-up.

MAPEBCDIC2ASCII=
  Specifies a translation table that is used by SAS users to transcode from
  EBCDIC to ASCII and from ASCII to EBCDIC.

ODSLANGCHG
  Specifies whether the language of the text of the ODS output can be changed.

LOCALEDATA
  Specifies the source database for the locale information.

TIMEZONE
  Specifies the user local time zone.

SAS 9.4M2 has the following new system option:

NLDECSEPARATOR
  Specifies whether SAS produces locale-sensitive numeric output for the decimal
  separator or continues to format numbers with U.S. English preferences.

The following system option was enhanced:

DFLANG
  Supports the locale option.

In SAS 9.4M2, the NLSCOMPATMODE system option was removed.

In SAS 9.4M3, the LINGUISTIC option was added to the SORTSEQ= system option
  on page 708.
PART 1

NLS Concepts

Chapter 1
National Language Support (NLS) .................................................. 3

Chapter 2
Locale for NLS ........................................................................... 5

Chapter 3
Encoding for NLS ...................................................................... 9

Chapter 4
Transcoding for NLS ................................................................. 31

Chapter 5
Double-Byte Character Sets (DBCS) ........................................... 45

Chapter 6
Time Zones ............................................................................. 51
Overview to National Language Support

National Language Support is a set of features that enable a software product to function properly in every global market for which the product is targeted. SAS contains NLS features to ensure that SAS applications can be written so that they conform to local language conventions. Typically, software that is written in the English language works well for users who speak the English language and use data that is formatted using the conventions that are observed in the United States. However, without NLS, these products might not work well for users in other regions of the world. NLS in SAS enables users in regions such as Asia and Europe to process data successfully in their native languages and environments.

NLS is applied to data that is moved between machines; for example, NLS ensures that the data is converted to the correct format for use on the target machine.

Text-string operations are sensitive to SAS settings for language and region. This action enables correct results for such operations as uppercase and lowercase characters, classifying characters, and scanning data. SAS provides features to ensure that national characters, which are characters specific to a particular nation or group of nations, are displayed and are printed properly.

Software applications that incorporate NLS can avoid dependencies on language-specific or cultural-specific conventions for software features. Here are some examples:

- string manipulation
- character classifications
- character comparison rules
- code sets
- date and time formatting
- user interfaces
- message-text languages
- numeric and monetary formatting
- sort orders
Definition of Localization and Internationalization

*Internationalization* is the process of designing a software product without making assumptions that are based on a single language or locale, thereby facilitating localization. Internationalization ensures that international conventions (including rules for sorting strings and for formatting dates, times, numbers, and currencies) are supported. It also facilitates a consistent user experience across different language editions of a product. The abbreviation for internationalization is I18N.

Although the application logic might support cultural conventions (for example, the monetary and numeric formats of a particular region), only a localized version of the software presents user interfaces and system messages in the local language.

*Localization* is the process of adapting a product to meet the language, cultural, and other requirements of a specific target environment or market so that users can see results created in their own languages and conventions when using the product. Translation of the user interface, system messages, and documentation is part of localization.

SAS NLS features are available for localizing and internationalizing your SAS applications. For more information see “Internationalization Compatibility for SAS String Functions” on page 343.
Locale for NLS

Overview of Locale Concepts for NLS

A locale reflects the language, local conventions such as data formatting, and culture for a geographical region. Local conventions might include specific formatting rules for dates, times, and numbers and a currency symbol for the country or region. Collating sequence, paper size, postal addresses, and telephone numbers can also be included in locale.

Dates have many representations, depending on the conventions that are accepted in a culture. The month might be represented as a number or as a name. The name might be fully spelled or abbreviated. The order of the month, day, and year might differ according to locale.

For example, “the third day of October in the year 2016” would be displayed in a different way for each of these locales:

Germany
03.10.16

Italy
3/10/16

United States
10/03/16

Language is part of a locale, but is not unique to any one locale. For example, Portuguese is spoken in Brazil as well as in Portugal, but the cultures are different. In Brazil and in Portugal, there are similarities in the formatting of data. Numbers are formatted using a comma (,) to separate integers from fractional values and a dot (.) to separate groups of digits to the left of the radix character. However, there are important differences, such as the currency symbols that are used in the two different locales. Portugal uses the Euro and requires the Euro symbol € while Brazil uses the Real that is represented by the two-character currency symbol R$.

Also, a country might have more than one official language. Canada has two official languages: English and French; two values can be specified for the LOCALE= system option: English_Canada and French_Canada.
Numbers, including currency, can have different representations. For example, the decimal separator, or radix character, is a dot (.) in some regions and a comma (,) in other regions. The thousands separator can be a dot, comma, or even a space in some regions. Monetary conventions likewise vary between locales; for example, a dollar sign or a yen sign might be attached to a monetary value.

Paper size and measurement are also locale considerations. Standard paper sizes include letter (8-1/2-by-11-inch paper) and A4 (210-by-297-millimeter paper). The letter paper size is mainly used by some English-speaking countries; A4 is used by most other locales. Most locales use centimeters, some locales use inches.

Time can be represented in one English-speaking country or region by using the 12-hour notation. Other English speakers expect time values to be formatted using the 24-hour notation.

Specifying a Locale

How Locale Is Specified at SAS Invocation

You can use the LOCALE= system option to specify the locale of the SAS session at SAS invocation. LOCALE= also implicitly sets the following SAS system options:

- DATESTYLE=
- DFLANG=
- ENCODING=
- LOCALEDATA
- MAPEBCDICTOASCII
- ODSLANGCHG
- PAPERSIZE=
- RSASIOTRANSERROR
- TIMEZONE
- URLENCODING

Note: Locale can also be specified using POSIX naming standards. For example, en_US is the POSIX equivalent for the SAS value English_UnitedStates.

Default values for the LOCALE= option are the same under each operating environment. For more information, see “LOCALE= Values for PAPERSIZE and DFLANG, Options” on page 811.

The English_UnitedStates value for LOCALE= causes the following options to be implicitly set to the specified Windows default values SAS invocation:

- DATESTYLE=MDY
- DFLANG=English
- ENCODING=WLATIN1
At invocation, an explicitly set system option overrides any implicitly set option.

At invocation, the explicit setting PAPERSIZE=A4 overrides an implicit setting of the PAPERSIZE= option using the LOCALE= option. For more information, see "PAGESIZE= System Option" in SAS System Options: Reference.

How Locale Is Specified during a SAS Session

You can use the LOCALE= system option to specify the locale of the SAS session during the SAS session. However, only the values for these system options change implicitly to reflect the changed value of LOCALE=:

- **DFLANG=**
- **PAPERSIZE=**

**Note:** The PAPERSIZE option is affected only if it is set to LOCALE. For more information see "PAGESIZE= System Option" in SAS System Options: Reference.

The value for the ENCODING= system option does not change implicitly to reflect the changed value of LOCALE=.

The Italian_Italy value that is assigned to the LOCALE= option implicitly resets the following options during the SAS session to reflect the changed value of the LOCALE= system option:

- **DFLANG=**Italian
- **PAPERSIZE=A4

Language Switching

SAS messages are displayed in the language that is specified by the settings in the SAS configuration file during start up. You can view SAS messages in another language by using the language switching feature. You can access the language switching feature with the ODSLANGCHG system option. If ODSLANGCHG is enabled, then the value of the LOCALE system option determines the language for procedure output, user interface elements, and ODS fonts. If ODSLANGCHG is disabled, then messages appear in the language that is set during start up. This feature is supported on any server as long as the characters of the messages are supported by the encoding. For more information, see the “ODSLANGCHG System Option” on page 706. The LSWLANG system option specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation. The LSWLANG system option has higher priority over the LOCALE system option. If the LSWLANG option is set to a valid SAS language, the SAS log output is controlled by its value. Otherwise, the LOCALE= option determines the language of the SAS log.

The LOGLANGCHG system option controls whether language switching occurs for the SAS log messages. This option controls the language of message switching in SAS log output. If LOGLANGCHG is specified, the language of the SAS log
depends on the LSWLANG or LOCALE= option. For more information, see “LOGLANGCHG System Option” on page 700.

The LSWLANG option specifies the language of messages if LOGLANGCHG or ODSLANGCHG are enabled. If LSWLANG is set to LOCALE, the LOCALE= option determines the language for switching. LSWLANG=LOCALE is the default. For more information, see “LSWLANG System Option” on page 703.

The LOGLANGENG option is a toggle option that overrides LOGLANGCHG and LSWLANG and sets them to LOGLANGCHG=ON and LSWLANG=EN. As a result, the LOG output is in English, and the NL format output does not change. This option changes the setting of ODSLANGCHG. If ODSLANGCHG=OFF, the system message language for ODS output is determined by SAS configuration. If ODSLANGCHG=ON, all messages are in English because of the LSWLANG setting. For more information, see “LOGLANGENG System Option” on page 701.

For more information about Language Switching, see Multilingual Computing with SAS® 9.4.
Overview: Encoding for NLS

An encoding maps each character in a character set to a unique numeric representation, which results in a table of all code points. This table is referred to as a code page, which is an ordered set of characters in which a numeric index (code point value) is associated with each character. The position of a character on the code page determines its two-digit hexadecimal number.

A character set is the set of characters and symbols that are used by a language or group of languages. A character set includes national characters (which are characters specific to 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.

An encoding method is a set of rules that assign the numeric representations to the set of characters. These rules govern the size of the encoding (number of bits used to store the numeric representation of the character) and the ranges in the code page where characters appear. The encoding methods result from the adherence to standards that have been developed in the computing industry. An encoding method is often specific to the computer hardware vendor.

An encoding results from applying an encoding method to a character set.

An individual character can occupy a different position in a code page, depending on the code page used. An example is the euro symbol, €. The Unicode code point for the euro symbol is U+20AC.

Each SAS session is set to a default encoding, which can be specified by using various SAS language elements.

---

### Difference between Encoding and Transcoding

Encoding establishes the default working environment for your SAS session. For example, the Windows LATIN1 encoding is the default encoding for a SAS session under Windows in a Western European locale such as the de_DE locale for German in Germany. For example, the Windows LATIN1 code point for the uppercase letter Ä is C4 hexadecimal.

**Note:** The default encoding varies according to the operating environment and the locale.

However, if you are working in an international environment (for example, you access SAS data that is encoded in German EBCDIC), the German EBCDIC code point for the uppercase letter Ä is 4A hexadecimal. In order for a version of SAS that normally uses Windows LATIN1 to properly interpret a data set that is encoded in German EBCDIC, the data must be transcoded. Transcoding is the process of converting data from one encoding to another. When SAS transcodes the Windows LATIN1 uppercase letter Ä to the German EBCDIC uppercase letter Ä, the hexadecimal representation for the character is converted from the value C4 to a 4A. For conceptual information, see Chapter 4, "Transcoding for NLS," on page 31.

---

### Common Encoding Methods

An encoding results from applying an encoding method to a coded character set.

An encoding method is the application of established industry rules to a coded character set to produce an encoded character scheme. Such rules prescribe the number of bits required for storing the numeric representation of a specific character and its code position in the encoding. ISO 2022 and UTF-8 are examples of
encoding methods. For more information, see “Standards Organizations for NLS Encodings” on page 14.

An encoding method is a set of rules that assign numeric representations to a set of characters. These rules govern the size of the encoding, the number of bits used to store the numeric representation of the characters, and the ranges in the encoding where characters appear.

Common encoding methods are listed here:

Unicode

Unicode is a common and popular character set. It provides up to 110,116 character combinations. Unicode can accommodate basically all of the world's languages. Unicode is a coded character set. The following are encodings of the Unicode character set:

UTF-8

is an MBCS encoding that contains the Latin-script languages, Greek, Cyrillic, Arabic, and Hebrew. It also includes East Asian languages such as Japanese, Chinese, and Korean. The characters in UTF-8 are of varying width, from 1 to 4 bytes. UTF-8 maintains ASCII compatibility by preserving the ASCII characters in code positions 1 through 128.

UTF-8 is the most common encoding. 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>
<thead>
<tr>
<th>Encoding</th>
<th>Languages</th>
<th>Storage Size Increase in UTF-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>English, Malay</td>
<td>0%</td>
</tr>
<tr>
<td>ISO-8859-1</td>
<td>Western European</td>
<td>10%</td>
</tr>
<tr>
<td>ISO-8859-7, plain text</td>
<td>Greek</td>
<td>90%</td>
</tr>
<tr>
<td>ISO-8859-7, 50% markup</td>
<td>Greek</td>
<td>45%</td>
</tr>
<tr>
<td>TIS-620, plain text</td>
<td>Thai</td>
<td>190%</td>
</tr>
<tr>
<td>TIS-620, 50% markup</td>
<td>Thai</td>
<td>95%</td>
</tr>
<tr>
<td>EUC-KR, plain text</td>
<td>Korean</td>
<td>50%</td>
</tr>
<tr>
<td>EUC-KR, 50% markup</td>
<td>Korean</td>
<td>55%</td>
</tr>
</tbody>
</table>

Footnote: Adapted from W3C Internationalization (I18n) Activity

UTF-16

is a 16-bit form that contains all of the most common characters in all modern writing systems. Most of the characters are uniformly represented with 2 bytes, although there is extended space, called surrogate space, for additional characters that require 4 bytes.
UTF-32
is a 32-bit form whose characters each occupy 4 bytes.

ASCII (American Standard Code for Information Interchange)
is a 7-bit encoding for the United States that provides 128 character combinations. The encoding contains characters for uppercase and lowercase English, American English punctuation, base 10 numbers, and a few control characters. This set of 128 characters is common to most other encodings. ASCII is used by personal and UNIX computers.

EBCDIC (Extended Binary Coded Decimal Interchange Code) family
is an 8-bit encoding that provides 256 character combinations. There are multiple EBCDIC-based encodings. EBCDIC is used on IBM mainframes and most IBM mid-range computers. EBCDIC follows ISO 646 conventions to facilitate translations between EBCDIC encodings and 7-bit (and 8-bit) ASCII-based encodings. The 95 EBCDIC graphical characters include 82 invariant characters (including a blank space), which occupy the same code positions across most EBCDIC single-byte code pages. It also includes 13 variant graphic characters, which occupy varying code positions across most EBCDIC single-byte code pages. For more information about variant characters, see “Code Point Discrepancies among EBCDIC Encodings” on page 14.

There are also multi-byte character set (MBCS) EBCDIC encodings.

ISO (International Organization for Standardization) 646 family
is a 7-bit encoding that is an international standard and provides 128 character combinations. The ISO 646 family of encodings is similar to ASCII except that it has 12 code points for national variants. The 12 national variants represent specific characters that are needed for a particular language.

ISO 8859 family and Windows family
is an 8-bit extension of ASCII that supports all of the ASCII code points and adds 12 more, providing 256 character combinations. LATIN1, which is officially named ISO-8859-1, is the most frequently used member of the ISO 8859 family of encodings. In addition to the ASCII characters, LATIN1 contains accented characters, other letters needed for languages of Western Europe, and some special characters. HTTP and HTML protocols are based on Unicode.

Other encodings
The ISO 8859 family has other members that are designed for other languages. The following table describes the other encodings that are approved by ISO.

<table>
<thead>
<tr>
<th>ISO Standard</th>
<th>Name of Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 8859-1</td>
<td>LATIN 1</td>
<td>US and Western European</td>
</tr>
<tr>
<td>ISO 8859-2</td>
<td>LATIN 2</td>
<td>Central and Eastern European</td>
</tr>
<tr>
<td>ISO 8859-3</td>
<td>LATIN 3</td>
<td>Southern European, Maltese, and Esperanto</td>
</tr>
<tr>
<td>ISO 8859-4</td>
<td>BALTIC</td>
<td>Northern European</td>
</tr>
</tbody>
</table>
A number of encoding standards have been developed for East Asian languages. Some of these are listed in the following table.

**Table 3.3 Some East Asian Language Encodings Approved by ISO**

<table>
<thead>
<tr>
<th>Standard</th>
<th>Name of Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB 2312-80</td>
<td>Simplified Chinese</td>
<td>People's Republic of China</td>
</tr>
<tr>
<td>CNS 11643</td>
<td>Traditional Chinese</td>
<td>Taiwan</td>
</tr>
<tr>
<td>BIG-5</td>
<td>Traditional Chinese</td>
<td>Taiwan</td>
</tr>
<tr>
<td>KS C 5601</td>
<td>Korean National Standard</td>
<td>Korea</td>
</tr>
<tr>
<td>JIS</td>
<td>Japan Industry Standard</td>
<td>Japan</td>
</tr>
<tr>
<td>SHIFT-JIS</td>
<td>Japan Industry Standard</td>
<td>Japan multibyte encoding</td>
</tr>
</tbody>
</table>

There are other encodings in the standards for EBCDIC and Windows that support different languages and locales.
Standards Organizations for NLS Encodings

Encodings that are supported by SAS are defined by the following standards organizations:

International Organization for Standardization (ISO)
promotes the development of standardization and related activities. ISO also establishes standards for encodings.

American National Standards Institute (ANSI)
coordinates voluntary standards and conformity to those standards in the United States. ANSI works with ISO to establish global standards.

Unicode Consortium
that develops and promotes the Unicode standard, which provides a unique number for every character.

Code Point Discrepancies among EBCDIC Encodings

The following characters are considered variant because they can have different code positions in various EBCDIC variations: ! # $ @ \ [ ] ^ ` { } | ~ These characters exist in every encoding, but their hexadecimal values might change from one encoding to another, as shown in the following table.

<table>
<thead>
<tr>
<th>Character</th>
<th>1047</th>
<th>838</th>
<th>870</th>
<th>1025</th>
<th>1141</th>
<th>1142</th>
<th>1143</th>
<th>1144</th>
<th>1145</th>
<th>1146</th>
<th>1147</th>
<th>1148</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>5A</td>
<td>5A</td>
<td>4F</td>
<td>4F</td>
<td>4F</td>
<td>4F</td>
<td>4F</td>
<td>BB</td>
<td>5A</td>
<td>4F</td>
<td>4F</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>7B</td>
<td>7B</td>
<td>7B</td>
<td>7B</td>
<td>4A</td>
<td>63</td>
<td>B1</td>
<td>69</td>
<td>7B</td>
<td>B1</td>
<td>7B</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>5B</td>
<td>5B</td>
<td>5B</td>
<td>5B</td>
<td>67</td>
<td>67</td>
<td>5B</td>
<td>5B</td>
<td>4A</td>
<td>5B</td>
<td>5B</td>
<td></td>
</tr>
<tr>
<td>@</td>
<td>7C</td>
<td>7C</td>
<td>7C</td>
<td>7C</td>
<td>B5</td>
<td>80</td>
<td>EC</td>
<td>B5</td>
<td>7C</td>
<td>7C</td>
<td>44</td>
<td>7C</td>
</tr>
<tr>
<td>\</td>
<td>E0</td>
<td>E0</td>
<td>E0</td>
<td>E0</td>
<td>EC</td>
<td>E0</td>
<td>71</td>
<td>48</td>
<td>E0</td>
<td>E0</td>
<td>48</td>
<td>E0</td>
</tr>
<tr>
<td>[</td>
<td>AD</td>
<td>49</td>
<td>4A</td>
<td>63</td>
<td>9E</td>
<td>B5</td>
<td>90</td>
<td>4A</td>
<td>B1</td>
<td>90</td>
<td>4A</td>
<td></td>
</tr>
<tr>
<td>]</td>
<td>BD</td>
<td>59</td>
<td>5A</td>
<td>5A</td>
<td>FC</td>
<td>9F</td>
<td>9F</td>
<td>51</td>
<td>5A</td>
<td>BB</td>
<td>B5</td>
<td>5A</td>
</tr>
</tbody>
</table>
Examples of characters that are commonly used in programming languages are { and $.

These characters are known as variant characters. For example, if a German mainframe user entered an ä, which occupies code point C0, an American compiler would interpret code point C0 as a {.

### Collating Sequence

#### Overview of Collating Sequence

The *collating sequence* is the order in which characters are sorted. For example, when the SORT procedure is executed, the collating sequence determines the sort order (higher, lower, or equal to) of a particular character in relation to other characters.

The default collating sequence is binary collation, which sorts characters according to each character’s location in the code page of the session encoding. (The session encoding is the default encoding for a SAS session. The default encoding can be specified by using various SAS language elements.) The sort order corresponds directly to the arrangement of the code points within the code page.

Binary collation is the fastest type of collation because it is the most efficient for the computer. However, locating characters within a binary-collated report might be difficult if you are not familiar with this method. For example, a binary-collated report lists words beginning with uppercase characters separately from words beginning with lowercase characters. It lists words beginning with accented characters after words beginning with unaccented characters. Therefore, for ASCII-based encodings, the capital letter Z precedes the lowercase letter a. Similarly, for EBCDIC-based encodings, the lowercase letter z precedes the capital letter A.

You can request an alternate collating sequence that overrides the binary collation. To request an alternate collating sequence, specify one of the following sequences:

- a translation table name
This example illustrates the results of using different collating sequences to sort a short list of words:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Translation Table</th>
<th>Encoding Value</th>
<th>Linguistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aaron</td>
<td>aardvark</td>
<td>Aaron</td>
<td>aardvark</td>
</tr>
<tr>
<td>Aztec</td>
<td>azimuth</td>
<td>Aztec</td>
<td>Aaron</td>
</tr>
<tr>
<td>Zeus</td>
<td>Aaron</td>
<td>Zeus</td>
<td>azimuth</td>
</tr>
<tr>
<td>aardvark</td>
<td>Aztec</td>
<td>aardvark</td>
<td>Aztec</td>
</tr>
<tr>
<td>azimuth</td>
<td>cote</td>
<td>azimuth</td>
<td>cote</td>
</tr>
<tr>
<td>cote</td>
<td>coté</td>
<td>cote</td>
<td>coté</td>
</tr>
<tr>
<td>coté</td>
<td>côté</td>
<td>coté</td>
<td>coté</td>
</tr>
<tr>
<td>côté</td>
<td>coté</td>
<td>côté</td>
<td>côté</td>
</tr>
<tr>
<td>côté</td>
<td>zebra</td>
<td>côté</td>
<td>zebra</td>
</tr>
<tr>
<td>zebra</td>
<td>zèbre</td>
<td>zebra</td>
<td>zèbre</td>
</tr>
<tr>
<td>zèbre</td>
<td>Zeus</td>
<td>zèbre</td>
<td>Zeus</td>
</tr>
</tbody>
</table>

The first column shows the results of binary collation on characters that are represented in an ASCII-based encoding. The alphabetization is not consistent because of the separate grouping of words that begin with uppercase and lowercase characters. For example, the word Zeus appears before aardvark because of the code points that are assigned to the characters within the ASCII-based encoding.

The second column shows the results of specifying a translation table that alternates the ordering of lowercase and uppercase characters. If you use the translation table, the word aardvark appears before Zeus. However, the word azimuth appears before Aaron because the translation table assigns a weight value to the lowercase character a that is less than the weight value of the uppercase character A. In addition, accents are sorted from left to right. For example, coté comes before côté.

The third column shows the results of specifying the ASCII-based, double-byte latin1 encoding.

The last column shows the results of linguistic collation for the session locale fr_FR (French_France), which uses a collation algorithm to alphabetize words. The algorithm specifies that words beginning with lowercase characters appear before...
words beginning with uppercase characters. In addition, this linguistic collation sorts accents from right to left because of the French locale specification.

SAS has adopted the International Components for Unicode (ICU) to implement linguistic collation. The ICU and its implementation of the Unicode Collation Algorithm (UCA) have become a standard. The collating sequence is the default provided by the ICU for the specified locale.

### Request Alternate Collating Sequence

To request an alternate collating sequence, use the following SAS language elements:

- **SORTSEQ= option in the PROC SORT statement.** See “Collating Sequence Option Statement” on page 723.
- **SORTSEQ= system option.** See “SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 708.

Note that neither method supports all of the collating sequences. For example, only the SORTSEQ= option in the PROC SORT statement supports linguistic collation. However, both the SORTSEQ= option in the PROC SORT statement and the SORTSEQ= system option support translation table collating sequences.

The BASE (V9) engine and the REMOTE engine for SAS/SHARE support all alternate collating sequences. The V9TAPE sequential engine supports the use of a translation table and an encoding value to sort data, but the V9TAPE engine does not support linguistic collation.

### Specifying a Translation Table

A translation table is a SAS catalog entry that transcodes data from one single-byte encoding to another single-byte encoding. A translation table also reorders characters when sorting them. A translation table can be one that SAS provides, such as a standard collating sequence like ASCII, EBCDIC, or DANISH; or it can be a user-defined translation table.

When you specify a translation table for an alternate collating sequence, the characters are reordered by mapping the code point of each character to an integer weight value in the range of 0 to 255. A binary collation is then performed.

For collating purposes, you can create translation tables that order characters so that lowercase and uppercase characters alternate. For example, you can create a translation table to correct the situation in which Z precedes A in an ASCII-based encoding. (However, regardless of the weight assignments in the translation table, it is difficult to achieve a true alphabetic ordering that takes the character case into account.) You can also create a translation table that orders alphabetic characters of a particular language in their expected order.

The TRANTAB procedure creates, edits, and displays translation tables. For example, you can display a translation table to view the character-weight values. The translation tables that are supplied by SAS are stored in the SASHELP.HOST catalog. Any translation table that you create or customize is stored in your SASUSER.PROFILE catalog. Translation tables have an entry type of TRANTAB.
See Chapter 20, "TRANTAB Procedure," on page 783 for more information about translation tables.

You can specify a translation table with the SORTSEQ= option in the PROC SORT statement or with the SORTSEQ= system option. For example, if your operating environment sorts with the ASCII-based Viatin1 encoding by default, and you want to sort with a translation table that alternates uppercase and lowercase characters, issue the following statements to specify the SAS translation table FRSOLAT1:

```sas
proc sort data=myfiles.test sortseq=FRSOLAT1;
   by name;
run;
```

A SAS data set that is sorted with a translation table contains a sort indicator that displays the specified translation table name as the collating sequence in CONTENTS procedure output.

---

### Specifying an Encoding Value

An *encoding* is a set of characters (letters, logograms, digits, punctuation marks, symbols, and control characters) that have been mapped to hexadecimal values, called code points, that computers use. When you specify an encoding value for an alternate collating sequence, the characters are transcoded from the SAS session encoding to the specified encoding. Then a binary collation is performed. You can specify all encoding values that are supported by the ENCODING= option, including multi-byte encodings. Note that specifying a translation table can transcode data, but translation tables are limited to single-byte encodings.

You can specify an encoding value with the SORTSEQ= option in the PROC SORT statement, but you cannot specify an encoding value in the SORTSEQ= system option. For example, you want to sort a SAS data set and then transport it to a Japanese Windows environment. If your session encoding is ASCII-based and binary collation is in effect, you can issue the following statements to specify the ASCII-based double-byte encoding SHIFT-JIS:

```sas
proc sort data=myfiles.test sortseq='shift-jis';
   by name;
run;
```

Note that SAS checks the encoding value for any translation tables with the same name. If a translation table name exists, SAS uses the translation table.

A SAS data set that is sorted with an encoding value contains a sort indicator that displays the specified encoding value as the collating sequence in CONTENTS procedure output.

---

### Specifying Linguistic Collation

*Linguistic collation* sorts characters according to rules of language and produces results that are intuitive and culturally acceptable. The results are similar to the collation used in printed materials such as dictionaries, phone books, and book indexes. Linguistic collation is useful for generating reports or other data presentations and for achieving compatibility between systems.
SAS incorporates the International Components for Unicode (ICU). ICU is an open-source library that provides routines for linguistic collation that are compatible with the Unicode Collation Algorithm (UCA). The UCA is a standard by which Unicode strings can be compared and ordered.

To request linguistic collation, you must use the SORTSEQ= option in the PROC SORT statement because the SORTSEQ= system option does not support linguistic collation. For example, the following statements cause the SORT procedure to collate linguistically, in accordance with the French_France locale:

```plaintext
options locale=fr_FR;
proc sort data=myfiles.test sortseq=linguistic;
   by name;
run;
```

When linguistic collation is requested, SAS uses the default linguistic collation algorithm that is provided by the ICU for the SAS session locale. This algorithm reflects the language, local conventions such as data formatting, and culture for a geographical region. You can modify the algorithm by specifying options in parentheses following the LINGUISTIC keyword. For example, you can specify a different locale; you can specify the CASE_FIRST= option to collate lowercase characters before uppercase characters, or vice versa; and so on. Generally, it is not necessary to specify options, because the ICU associates defaults with the various languages and locales. For more information about the linguistic options, see the SORTSEQ= option in "Collating Sequence Option Statement" on page 723 or the SORTSEQ= option in the PROC SORT statement in Base SAS Procedures Guide.

A SAS data set that is sorted linguistically contains a sort indicator that displays the collating sequence LINGUISTIC in CONTENTS procedure output. Along with the sort indicator, the data set also records a complete description of the linguistic collating sequence in the file's descriptor information. The descriptor information is also displayed in CONTENTS procedure output.

## Determining the Encoding of a SAS Data Set

### Using PROC CONTENTS to Display the Encoding of a SAS Data Set

You can display the encoding of a data set by using PROC CONTENTS:

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

Here is the output from the PROC CONTENTS code. The encoding is US–ASCII.
Default SAS Session Encoding

The ENCODING= system option is used to specify the SAS session encoding. The session encoding establishes the environment to process SAS syntax and SAS data sets, and to read and write external files. If neither the LOCALE= nor ENCODING= options is set, a default value is set.

The ENCODING system option is set explicitly in all SASFoundation sasv9.cfg configuration files.

The BI server configuration process uses the default SASFoundation sasv9.cfg file (ISASROOT/sasv9.cfg). The default sasv9.cfg file points to the sasv9.cfg that contains the LOCALE and ENCODING option that you select. This action usually happens during deployment. You can deploy multiple server contexts at the site that uses different SAS configurations.

<table>
<thead>
<tr>
<th>Operating Environment</th>
<th>Default ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS</td>
<td>OPEN_ED-1047</td>
<td>OpenEdition EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>UNIX</td>
<td>Latin1</td>
<td>Western (ISO)</td>
</tr>
<tr>
<td>Windows</td>
<td>WLatin1</td>
<td>Western (Windows)</td>
</tr>
</tbody>
</table>

For a complete list of supported encoding values for a SAS session, see Chapter 24, “Encoding Values for a SAS Session,” on page 841.
Setting the Encoding of a SAS Session

You can set the session encoding by using the ENCODING= system option, the DBCS options, or the LOCALE= system option.

Note: Values for the ENCODING= system option depend on the operating environment.

The priority order for setting the encoding is as follows:

1 ENCODING= system option

The SAS session encoding is determined by the ENCODING= option regardless of whether the DBCS or LOCALE= options are specified. If the ENCODING= option is specified, a set of valid DBCS options is set regardless of whether the user has specified those options. Also, if the ENCODING= option is specified, the LOCALE= option is set to an appropriate value unless a value has been specified by the user.

Note: If the ENCODING= option is specified, the TRANTAB= option is implicitly set. TRANTABS apply only to external files.

2 DBCS options

Most North and South American, European, Middle Eastern, African, and Australian users use the SAS SBCS environment and do not use the DBCS environment.

If the ENCODING= option is not specified, the SAS session encoding is determined by the DBCS options regardless of whether the LOCALE= option is specified. The LOCALE= option is set to an appropriate value unless a value has been specified by the user.

The encoding is determined by the values of the DBCSLANG and DBCSTYPE options for DBCS languages, such as Japanese, Korean, Simplified Chinese, and Traditional Chinese.

The DBCS options are valid only when the DBCS extension directory is included in the path option list. The path of the DBCS extension dynamic link library (DLLs) must be located at the top of the pathname list of the path option for the DBCS languages when you want to invoke a DBCS SAS session. The DBCS extension DLLs are located in the directory $SASROOT/dbcs/sasexe by default.

Also, you might have to specify the resourcesloc, msg, and sashelp options to use localized resources even if the SAS session encoding is not a DBCS language (for example, Polish, German, and French). The localized resources are located under $SASROOT/nls/<language identifier>/<sasmsg, sashelp, sasmacro, resource>. The values for language identifiers are: cs, de, en, es, fr, hu, it, ja, ko, pl, ru, sv, zh, and zt.
You can specify a `sasv9.cfg` file located in the localized directories such as `SASROOT/nls/<language identifier>` so that you do not have to consider using the `path`, `resourcesloc`, `sasmsg`, and `sashelp` options.

If DBCS (which specifies that SAS process DBCS encodings) is specified, `DBCSTYPE=` and `DBCSSLANG=` options are implicitly set. The default values for `DBCSTYPE=` and `DBCSSLANG=` match those values for the DBCS environment on the host (for example, Japanese, Korean, or Chinese).

### 3 LOCALE= system option

The SAS session encoding is determined by the `LOCALE=` option and the platform, if the `ENCODING=` or DBCS options are not specified.

The following example shows that encoding is explicitly set by default for the Spanish_Spain locale:

```
sas9 -locale Spanish_Spain
```

The WLATIN1 encoding is the default encoding for the Spanish_Spain locale.

The following example shows that the WLATIN2 encoding is set explicitly when SAS is invoked:

```
sas9 -encoding wlatin2
```

---

Note: Setting DBCS encodings, DBCS options, or a CJK (Chinese, Japanese, Korean) locale on SAS if the DBCS extensions are not available fails to successfully invoke SAS.

---

Note: Changing the encoding for a SAS session does not affect SAS keywords or SAS log output, which remain in English.

---

In Table 3.7 on page 22, the following values for the CJK locales are based on locale and platform:

**Table 3.7 Default Encoding Values Based on the LOCALE= Option**

<table>
<thead>
<tr>
<th>Locales</th>
<th>WIN</th>
<th>MVS</th>
<th>UNX</th>
</tr>
</thead>
<tbody>
<tr>
<td>zh_TW</td>
<td>MS-950 (ywin)</td>
<td>IBM-937 (yibm)</td>
<td>Solaris on X64, Solaris on SPARC, EUC-TW (yeuc) others: MS-950 (ywin)</td>
</tr>
<tr>
<td>zh_HK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zh_MO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zh_CN</td>
<td>EUC-CN (zeuc)</td>
<td>IBM-935 (zibm)</td>
<td>EUC-CN (zeuc)</td>
</tr>
<tr>
<td>zh_SG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ja_JP</td>
<td>SHIFT-JIS (sjis)</td>
<td>IBM-939 (jibm)</td>
<td>h64, h6i, AIX on Power, SHIFT-JIS (sjis) others: EUC-JP (jeuc)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM-930(j930)</td>
<td></td>
</tr>
</tbody>
</table>
Encoding Behavior in a SAS Session

Encoding Support for Data Sets by SAS Release

For Base SAS files, there are three categories of encoding support. Encoding support is based on the version of SAS that created the file:

- Data sets that are created in SAS 9 automatically have an encoding attribute, which is specified in the descriptor portion of the file. In SAS 9, DBCS by default recognizes the DBCSTYPE value and converts it to the encoding value and specifies it in the descriptor portion of the field.

- Data sets that are created in SAS 7 and SAS 8 do not have an encoding value that is specified in the file. It is assumed that SAS 7 and SAS 8 data sets were created in the SAS session encoding of the operating environment. However, the descriptor portion of the file does support an encoding value. When you replace or update a SAS 7 or SAS 8 file in a SAS 9 session, SAS specifies the current session encoding in the descriptor portion of the file, by default. In SAS 8, DBCS has the DBCSTYPE field, instead of the encoding field.

- Data sets created in SAS 6 do not have an encoding value that is associated with the file. An encoding value cannot be specified in the file.

Output Processing

When you create a data set in SAS 9, encoding is determined as follows:

- If a new output file is created, the data is written to the file using the current session encoding.

- If a new output file is created using the OUTREP= option, the data is written to the file using the default session encoding for the operating system that is specified by the OUTREP= value. The OUTREP= option specifies a data representation that is different from the current session. For more information, see “OUTREP= Data Set Option” on page 72.

- If a new output file replaces an existing file, the new file inherits the encoding of the existing file. For output processing that replaces an existing file that is from another operating environment or if the existing file has no encoding that is specified in it, then the current session encoding is used.
Input Processing

For input (read) processing in SAS 9, encoding behavior is as follows:

- Most users choose the default behavior that does not specify an encoding for the input file.
- If the session encoding and the encoding that is specified in the file are incompatible, the data is transcoded to the session encoding. For example, if the current session encoding is ASCII and the encoding that is specified in the file is EBCDIC, SAS transcodes the data from EBCDIC to ASCII.
- If a file does not contain a specified encoding, SAS transcodes the data only if the file's data representation is different from the session encoding.

Reading and Writing External Files

SAS reads and writes external files using the current session encoding. SAS assumes that the external file has the same encoding as the session encoding. For example, if you are creating a new SAS data set by reading an external file, SAS assumes that the encoding of the external file and the current session are the same. If the encodings are not the same, the external data could be written incorrectly to the new SAS data set. You need to specify an appropriate ENCODING option. The following example specifies the Shift-JIS encoding:

```sas
filename in 'external-file'
  encoding='Shift-JIS';
data mylib.contacts;
infile in;
length name $ 30 first $ 30 street $ 60 zip $ 10 city $ 30;
input name first street zip city;
run;
```

For details about the syntax for the SAS statements that perform input and output processing, see “SAS Options That Transcode SAS Data” on page 34.

Migrating Data to UTF-8

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

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. There are several methods of migrating data: Cross-environment data access (CEDA), Character Variable Padding (CVP) engine, and the %COPY_TO_NEW_ENCODING macro. Instructions on how to use these methods are described in Migrating Data to UTF-8 for SAS Viya.
Migrating Data from WLATIN1 to UTF-8

The common encodings for Western European character data are ISO 8859-1 (LATIN1) and Windows cp1252 (WLATIN1). If your character data is encoded as LATIN1 or WLATIN1, there are several factors that you must consider when you migrate to UTF-8. This section focuses on WLATIN1 characters, which are a superset of LATIN1 characters.

All WLATIN1 characters are available in UTF-8. Here is an example of the WLATIN1 code page:

Figure 3.1  WLATIN1 Code Page
Characters in WLATIN1 require 1 byte, and UTF-8 characters require 1–4 bytes. In Figure 3.2, the characters in the highlighted rows 2–7 require 1 byte in UTF-8. These characters are also referred to as ASCII characters.

Figure 3.2  Rows 2–7 in the WLATIN1 Code Page

In Figure 3.3, the characters in the highlighted rows 8–F require 1 byte in WLATIN1 and 2–3 bytes in UTF-8.
For errors or incorrect data that result from migrating characters that are 1 byte in WLATIN1 but 2–4 bytes in UTF-8, follow these recommendations to resolve the problem:

**Truncation**
All WLATIN1 characters can be transcoded to UTF-8. A transcoding error or warning means that the character variable is not long enough to hold the UTF-8 representation of those characters. To resolve the problem, you can expand the size of your character columns using **CVP**.

**Unexpected characters**
If you are expecting all characters in your WLATIN1 data to be 1 byte in UTF-8, you might be surprised to see a transcoding error when you migrate your data to UTF-8. The WLATIN1 encoding includes two versions of some punctuation characters:
- ASCII version in rows 2–7
Some text editors, such as Microsoft Word, support an AutoFormat feature that converts the ASCII characters that are entered from the keyboard to a version of the character that is produced by a typographical device.

For example, the quotation mark character (”) (also known as a straight quotation mark) has a hexadecimal representation of 22 in both WLATIN1 and UTF-8. However, AutoFormat might convert the character to left and right double quotation characters (also referred to as curly quotation marks or smart quotation marks). The hexadecimal values for the left and right double quotation marks in WLATIN1 are 93 and 94. But, in UTF-8, these characters are E2809C and E2809D, and each character requires 3 bytes.

You can use the **KPROPDATA function** with the PUNC option to convert smart quotation marks and other typographical punctuation characters to a 1-byte character in UTF-8.

Here are the punctuation characters that are 1 byte in WLATIN1 and 2–4 bytes in UTF-8:

**Table 3.8 Smart Quotation Marks and Punctuation Characters**

<table>
<thead>
<tr>
<th>Category</th>
<th>Punctuation character</th>
<th>Description</th>
<th>WLATIN1 code point (Hex)</th>
<th>UTF-8 code point (Hex)</th>
<th>ASCII Character</th>
<th>Description</th>
<th>ASCII code point (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quotation marks</td>
<td>’</td>
<td>single low-9 quotation mark</td>
<td>82</td>
<td>E2 80 9A</td>
<td>,</td>
<td>comma</td>
<td>2C</td>
</tr>
<tr>
<td></td>
<td>“</td>
<td>double low-9 quotation mark</td>
<td>84</td>
<td>E2 80 9E</td>
<td>,,</td>
<td>quotation mark</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>’</td>
<td>left single quotation mark</td>
<td>91</td>
<td>E2 80 98</td>
<td>’</td>
<td>apostrophe</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>’</td>
<td>right single quotation mark</td>
<td>92</td>
<td>E2 80 99</td>
<td>’</td>
<td>apostrophe</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>“</td>
<td>left double quotation mark</td>
<td>93</td>
<td>E2 80 9C</td>
<td>”</td>
<td>quotation mark</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>”</td>
<td>right double quotation mark</td>
<td>94</td>
<td>E2 80 9D</td>
<td>”</td>
<td>quotation mark</td>
<td>22</td>
</tr>
<tr>
<td>Category</td>
<td>Punctuation character</td>
<td>Description</td>
<td>WLATIN1 code point (Hex)</td>
<td>UTF-8 code point (Hex)</td>
<td>ASCII Character</td>
<td>Description</td>
<td>ASCII code point (Hex)</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------</td>
<td>--------------------------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Angle brackets</td>
<td>‹</td>
<td>single left-pointing angle quotation mark</td>
<td>8B</td>
<td>E2 80 B9</td>
<td>&lt;</td>
<td>less-than sign</td>
<td>3C</td>
</tr>
<tr>
<td></td>
<td>›</td>
<td>single right-pointing angle quotation mark</td>
<td>9B</td>
<td>E2 80 BA</td>
<td>&gt;</td>
<td>greater-than sign</td>
<td>3E</td>
</tr>
<tr>
<td>Ellipsis</td>
<td>…</td>
<td>horizontal ellipsis</td>
<td>85</td>
<td>E2 80 A6</td>
<td>…</td>
<td>full stop</td>
<td>2E 2E 2E 1</td>
</tr>
<tr>
<td>Bullet</td>
<td>•</td>
<td>bullet</td>
<td>95</td>
<td>E2 80 A2</td>
<td>*</td>
<td>asterisk</td>
<td>2A</td>
</tr>
<tr>
<td>Hyphen</td>
<td>–</td>
<td>en hyphen</td>
<td>96</td>
<td>E2 80 93</td>
<td>-</td>
<td>hyphen-minus</td>
<td>2D</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>em hyphen</td>
<td>97</td>
<td>E2 80 94</td>
<td>-</td>
<td>hyphen-minus</td>
<td>2D</td>
</tr>
</tbody>
</table>

1 Three ASCII period characters.
Transcoding for NLS

Overview of Transcoding

Transcoding is the process of converting data from one encoding to another. Transcoding is necessary when the SAS session encoding and the encoding of the data are different. Transcoding is often necessary when you move data between operating environments that use different locales and encoding.

The sort order corresponds directly to the arrangement of the code points within the code page.

For example, consider a file that was created under a Linux operating environment that uses the LATIN1 encoding, then moved to an IBM mainframe that uses the German EBCDIC encoding. When the file is processed on the IBM mainframe, the data is remapped from the LATIN1 encoding to the German EBCDIC encoding. If the data contains an uppercase letter Ä, the hexadecimal number is converted from C4 to 4A.

Transcoding does not translate between languages; transcoding remaps characters.

To dynamically transcode data between operating environments that use different encodings, an explicit encoding value must be specified. For details, see Chapter 23, "Encoding Values in SAS Language Elements," on page 829.
Common Reasons for Transcoding

Some situations where data might commonly be transcoded are:

- sharing data between two different SAS sessions that are running in different locales or in different operating environments
- reading an external file and the file’s encoding is incompatible with the session encoding

Transcoding and Translation Tables

Translation tables are implicitly set by the LOCALE= system option. They are used only for transcoding external files. There is direct transcoding for SAS files between the session encodings.

Specifying LOCALE= or ENCODING= indirectly sets the appropriate translation table values in the TRANTAB= option. Translation tables are used for transcoding one SBCS encoding to another and back again. For example, there is a specific translation table that maps Windows LATIN2 to ISO LATIN2. Translation tables apply only to external files.

The following figure shows a translation table. The area of a translation table for mapping from Windows LATIN2 (wlt2) to ISO LATIN 2 (lat2) is named "table 1," and the area for mapping characters from ISO LATIN2 to Windows LATIN2 is named "table 2."
The LOCALE= or ENCODING= system option and other encoding options (statements, commands, or procedures) eliminate the need to directly create or manage translation tables.

**CAUTION**

Do not change a translation table unless you are familiar with its purpose.

Translation tables are used internally by the SAS supervisor to implement NLS. If you are unfamiliar with the purpose of translation tables, do not change the specifications without proper technical advice.

The TRANTAB= option specifies the translation table to be used in the SAS session. For details, see “TRANTAB= System Option” on page 711. The TRANTAB procedure is used to create, edit, and display customized translation tables. For details, see Chapter 20, “TRANTAB Procedure,” on page 783.
SAS Options That Transcode SAS Data

The following SAS options for various language elements enable you to transcode, or to override the default encoding behavior. These elements enable you to specify a different encoding for a SAS file or a SAS application or to suppress transcoding.

Table 4.1  SAS Options That Transcode SAS Data

<table>
<thead>
<tr>
<th>Option</th>
<th>Where Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARSET=</td>
<td>ODS MARKUP statement</td>
</tr>
<tr>
<td>CORRECTENCODING=</td>
<td>MODIFY statement of the DATASETS procedure</td>
</tr>
<tr>
<td>ENCODING=</td>
<td>%INCLUDE, FILE, FILENAME, INFILE, ODS statements; FILE and INCLUDE commands</td>
</tr>
<tr>
<td>ENCODING=</td>
<td>in a DATA step</td>
</tr>
<tr>
<td>INENCODING=</td>
<td>LIBNAME statement</td>
</tr>
<tr>
<td>ODSCHARSET=</td>
<td>LIBNAME statement for XML</td>
</tr>
<tr>
<td>ODSTRANTAB=</td>
<td>LIBNAME statement for XML</td>
</tr>
<tr>
<td>OUTENCODING=</td>
<td>LIBNAME statement</td>
</tr>
<tr>
<td>XMLENCODING=</td>
<td>LIBNAME statement for XML</td>
</tr>
</tbody>
</table>

For a list of supported encoding values to use for these options, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

Transcoding Considerations

Although transcoding usually occurs with no problems, there are situations that can affect your data and produce unsatisfactory results. Here are several examples:

- One encoding can conflict with another. That is, two encodings can use different code points for the same character, or use the same code points for two different characters.
Characters in one encoding might not be present in another encoding. For example, a specific encoding might not have a character for the dollar sign ($). Transcoding the data to an encoding that does not support the dollar sign would result in the character not printing or displaying.

The number of bytes for a character in one encoding can be different from the number of bytes for the same character in another encoding. An example is transcoding from a double-byte character set (DBCS) to a single-byte character set (SBCS). Therefore, transcoding can result in character value truncation.

If an error occurs during transcoding so that the data cannot be transcoded back to its original encoding, data can be lost. That is, if you open a data set for update processing, the observation might not be updated. However, if you open the data set for input (read) processing and no output data set is open, SAS issues a warning that can be printed. Processing proceeds and allows a PRINT procedure or other Read operation to show the data that does not transcode.

CEDA has some processing limitations. For example, CEDA does not support update processing. See “Definitions for Cross-Environment Data Access (CEDA)” in SAS Programmer’s Guide: Essentials.

Incorrect encoding can be stamped on a SAS 7 or SAS 8 data set if it is copied or replaced in a SAS 9 session with a different session encoding from the data. The incorrect encoding stamp can be corrected with the CORRECTENCODING= option in the MODIFY statement in PROC DATASETS. If a character variable contains binary data, transcoding might corrupt the data.

Data set and variable labels have a maximum of 256 bytes and variable names have a maximum of 32 characters. If a data set contains long labels with international characters, the transcoded labels might exceed the maximum, resulting in a transcoding error. You can shorten the long labels in the original data set to resolve the transcoding issue.

Compatible and Incompatible Encodings

Overview of Compatible and Incompatible Encodings

An incompatible file is either a file that has a different data representation from the host architecture or an encoding value that differs from the SAS session encoding.

Table 4.2 on page 37 lists encodings that do not require CEDA processing for transcoding. For example, the first row is for the encoding method SBCS ASCII. If the session encoding is LATIN1, and the data set’s encoding is LATIN9, CEDA processing does not occur. Another example is the DBCS ASCII encoding method. Any combination of encodings in the first line of that row, SHIFT-JIS, MS-932, IBM-942, MACOS-1 does not invoke CEDA processing. If the session encoding is IBM-942, and the data set’s encoding is MACOS-1, CEDA processing does not occur.

In order to handle character data correctly, SAS requires that data be encoded using the session encoding. When SAS reads a data set that has a different encoding, CEDA compares the data set encoding with the SAS session encoding. The encodings might be incompatible, meaning that transcoding is required in order for SAS to safely handle the characters that are read from the data set.

ASCII is the foundation for many encodings, and it is used by most personal computers and workstations. However, the IBM mainframe uses an EBCDIC encoding. ASCII and EBCDIC machines, and the encodings that are used to store their data, are incompatible. Therefore, transcoding is required. For example, data must be transcoded if the SAS file is WLatin1 and the session encoding is OPEN_ED-1047. Note that SAS CEDA processing is used only in a mainframe environment when the data sets reside in a hierarchical file system such as z/FS or USS. Encodings that support characters for different languages are also incompatible. For example, WLatin1 is a single-byte (SBCS) encoding that supports characters for Western European languages. The WCYRILLIC is also SBCS, but it has Russian characters and does not have many of the Western European characters. Therefore, WLatin1 and WCYRILLIC are incompatible.

You can disable CEDA transcoding. For example, if the data set encoding is not compatible with the session encoding, but you know that all of the characters in the data set are compatible, transcoding occurs even though it is not necessary. You might prefer to transcode the data using a function such as KPROPDATA or KCVT instead of CEDA transcoding the data. You can prevent CEDA from transcoding by specifying one of these encoding names in either the ENCODING= data set option, or in either the INENCODING= or OUTENCODING= option in the LIBNAME statement.

**CAUTION**

Misuse of these encoding values could result in data corruption. ASCIIANY prevents transcoding.

- ANY specifies that no transcoding is desired, even between EBCDIC and ASCII encodings.

  **Note:** ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

- ASCIIANY enables you to create a data set that is compatible with all ASCII-based encodings. The encoding value that is set for the SAS data set using ASCIIANY is US-ASCII.

- EBCDICANY enables you to create a data set that is compatible with all EBCDIC-based encodings.

It is important for you to know the data in your data set before using one of these encodings to disable transcoding. For example, if the encoding of your data set is WLatin1 and the session encoding is WLatin2, CEDA transcodes the data because the encodings are not compatible. However, if you know that all of the characters in the data set are ASCII characters, specify ASCIIANY as the INENCODING option in the LIBNAME statement to prevent transcoding.

**Note:** If you are running SAS in a UTF-8 session encoding, any data set that is not UTF-8 or US-ASCII is incompatible.

```
libname myfiles 'path to data sets' inencoding=asciiany;
```
The encoding of the data set is not changed, but CEDA assumes that the data set encoding is the encoding that is specified in the LIBNAME statement.

As an alternative, you can prevent transcoding by changing the encoding in the data set header. To do so, use the CORRECTENCODING option in the PROC DATASETS MODIFY statement.

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

You can use these functions to verify that encodings are compatible:

- The ENCODCOMPAT function verifies the transcoding compatibility between two encodings. For more information, see “ENCODCOMPAT Function” on page 394.

- The ENCODISVALID function specifies a valid encoding name. For more information, see “ENCODCOMPAT Function” on page 394.

For a list of the encodings, by operating environment, see Chapter 24, “Encoding Values for a SAS Session,” on page 841.

### Table 4.2 Encoding Combinations That Do Not Need CEDA Processing for Transcoding

<table>
<thead>
<tr>
<th>Encoding Method</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCS ASCII</td>
<td>LATIN1, LATIN9</td>
</tr>
<tr>
<td></td>
<td>LATIN1, WLATIN1</td>
</tr>
</tbody>
</table>

**TIP**  
Transcoding is not performed in SAS using CEDA between these two encodings. WLATIN1 differs from LATIN1 by using displayable characters rather than control characters in the 80 to 9F (hexadecimal) range. These differences can lead to problems when cutting and pasting from MS WORD documents.
<table>
<thead>
<tr>
<th>Encoding Method</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCS EBCDIC</td>
<td>EBCDIC037, OPEN_ED-037</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1025, OPEN_ED-1025</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1026, OPEN_ED-1026</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1047, OPEN_ED-1047</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1112, OPEN_ED-1112</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1122, OPEN_ED-1122</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1130, OPEN_ED-1130</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1140, OPEN_ED-1140</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1141, OPEN_ED-1141</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1142, OPEN_ED-1142</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1144, OPEN_ED-1144</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1145, OPEN_ED-1145</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1146, OPEN_ED-1146</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1147, OPEN_ED-1147</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1148, OPEN_ED-1148</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1160, OPEN_ED-1160</td>
</tr>
<tr>
<td></td>
<td>EBCDIC1164, OPEN_ED-1164</td>
</tr>
<tr>
<td></td>
<td>EBCDIC037, OPEN_ED-037</td>
</tr>
<tr>
<td></td>
<td>EBCDIC275, OPEN_ED-275</td>
</tr>
<tr>
<td></td>
<td>EBCDIC424, OPEN_ED-424</td>
</tr>
<tr>
<td></td>
<td>EBCDIC425, OPEN_ED-425</td>
</tr>
<tr>
<td></td>
<td>EBCDIC838, OPEN_ED-838</td>
</tr>
<tr>
<td></td>
<td>EBCDIC870, OPEN_ED-870</td>
</tr>
<tr>
<td></td>
<td>EBCDIC875, OPEN_ED-875</td>
</tr>
<tr>
<td></td>
<td>EBCDIC924, OPEN_ED-924</td>
</tr>
</tbody>
</table>
### Encoding Method

<table>
<thead>
<tr>
<th>Encoding Method</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBCS ASCII</strong></td>
<td>SHIFT-JIS, MS-932, IBM-942, MACOS-1</td>
</tr>
<tr>
<td></td>
<td>EUC-KR, MS-949, MACOS-3</td>
</tr>
<tr>
<td></td>
<td>EUC-CN, MS-936, DEC-CN, MACOS-25</td>
</tr>
<tr>
<td></td>
<td>EUC-TW, DEC-TW, EUC-TW</td>
</tr>
<tr>
<td></td>
<td>BIG5, MACOS-2, MS-950</td>
</tr>
<tr>
<td><strong>DBCS EBCDIC</strong></td>
<td>IBM-930, OPEN_ED-930</td>
</tr>
<tr>
<td></td>
<td>IBM-933, OPEN_ED-933</td>
</tr>
<tr>
<td></td>
<td>IBM-935, OPEN_ED-935</td>
</tr>
<tr>
<td></td>
<td>IBM-937, OPEN_ED-937</td>
</tr>
<tr>
<td></td>
<td>IBM-939, OPEN_ED-939</td>
</tr>
<tr>
<td></td>
<td>IBM-939E, OPEN_ED-939E</td>
</tr>
<tr>
<td></td>
<td>IBM-930E, OPEN_ED930E</td>
</tr>
</tbody>
</table>

### Session Encodings and Data Set Encodings, Notes, Warnings, and Errors

The encoding used to create a SAS data set is saved in the data set header. For example, if you create a data set when you are running SAS with a session encoding of WLatin1, the WLatin1 encoding is stored in the data set header. SAS expects the character data encoding to be the same as the encoding in your SAS session. When the encodings do not match, CEDA must be used to transcode the data.

If the data set encoding and session encoding do not match, and the message is a NOTE, the character data should be valid. For example, when a UTF-8 session reads a data set created with a WLatin1 encoding, you will see the following note in your log:

```plaintext
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 the data set encoding and the session encoding are not compatible, transcoding fails, and the message is a WARNING or an ERROR:
This error can mean there is not enough space in one or more character columns in the data set's observation buffer to convert the data to the session encoding or that the characters can't be represented in the session encoding.

If the data set contains characters that are not supported by the session encoding, the message means that characters were lost during transcoding. For example, if the session encoding is WLATIN1, which supports characters for Western European languages, and SAS reads a data set that is SHIFT-JIS and contains Japanese characters, the message in the SAS log means that the Japanese characters are lost during transcoding.

Line-feed Characters and Transferring Data between EBCDIC and ASCII

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

- The encodings that have EBCDIC in their names use the traditional mapping of EBCDIC line-feed to ASCII line-feed character, which can cause data to appear as one stream.
- The encodings that have Open Edition in their names use the line-feed character as the end-of-line character. When the data is transferred to an operating environment that uses ASCII, the EBCDIC newline character maps to an ASCII line-feed character. This mapping enables ASCII applications to interpret the end-of-line correctly, resulting in better formatting.

For a list of the encodings, by operating environment, see Chapter 24, “Encoding Values for a SAS Session,” on page 841.

EBCDIC and OpenEdition Encodings Are Compatible

EBCDIC and OpenEdition are compatible encodings.

Encodings that contain EBCDIC in their names use the traditional mapping of EBCDIC line-feed (0x25) and newline (0x15) characters.

Encodings that contain OPEN_ED in their names and OpenEdition in their descriptions switch the mapping of the newline and line-feed characters. That is, they use the line-feed character as the end-of-line character.

If the two encodings use the same code page number but one is EBCDIC and the other is Open Edition, no transcoding is necessary.

Example:
If the data is encoded in EBCDIC1143 and the SAS session is encoded in OPEN_ED-1143, no transcoding is necessary because they use the same 1143 code page.

In order to transfer data between ASCII and EBCDIC, you can specify Open Edition encodings from the list of compatible encodings.

Encoding Compatibility in the z/OS Environment

Different encodings affect SAS applications at different release levels in the z/OS environment. You can migrate your SAS applications so that they use NONLSCOMPATMODE. See Ensuring Compatibility of Encodings.

Avoiding Character Data Truncation By Using the CVP Engine

If your data set encoding does not match the encoding of your SAS session, the character data in your data set might need more space in the new encoding. For example, if your SAS session is using a UTF-8 session encoding, and you are reading a data set with an encoding of Windows cp 1252 (WLATIN1), some of the characters that require one byte in WLATIN1 might require 2 or 3 bytes in UTF-8. To avoid data truncation, your character variables must expand to a width that is 1.5 times greater than the width of the original data.

When you process a SAS data file that requires transcoding, you can request that the CVP (character variable padding) engine expand character variable lengths so that character data truncation does not occur. (A variable’s length is the number of bytes used to store each of the variable’s values.)

Character data truncation can occur when the number of bytes for a character in one encoding is different from the number of bytes for the same character in another encoding, such as when a single-byte character set (SBCS) is transcoded to a double-byte character set (DBCS) or to a multi-byte character set (MBCS). An SBCS represents each character in one byte, and a DBCS represents each character in one or two bytes. An MBCS represents characters in a varying length from one to four bytes. For example, when transcoding from WLATIN1 to a Unicode encoding, such as UTF-8, the variable lengths (in bytes) might not be sufficient to hold the values, and the result is character data truncation.

Using the CVP engine, you specify an expansion amount so that variable lengths are expanded before transcoding, then the data is processed. Think of the CVP engine as an intermediate engine that is used to prepare the data for transcoding. After the lengths are increased, the primary engine, such as the default base engine, is used to do the actual file processing.

The CVP engine is a read-only engine for SAS data files only. You can request character variable expansion (for example, with the LIBNAME statement) in either of the following ways:

- explicitly specify the CVP engine and using the default expansion of 1.5 times the variable lengths.
implicitly specifying the CVP engine with the LIBNAME options CVPBYTES= or CVPMULTIPLIER=. These options specify the expansion amount. In addition, you can use the CVPPRINT= option to specify the primary engine to use for processing the SAS file; the default is the default SAS engine.

For example, the following LIBNAME statement explicitly assigns the CVP engine. Character variable lengths are increased using the default expansion, which multiplies the lengths by 1.5. For example, a character variable with a length of 10 has a new length of 15, and a character variable with a length of 100 has a new length of 150:

libname expand cvp 'SAS data-library';

Note: The expansion amount must be large enough to accommodate any expansion. Otherwise, truncation still occurs. An error is written to the SAS log.

Note: For processing that conditionally selects a subset of observations by using a WHERE expression, using the CVP engine might affect performance. Processing the file without using the CVP engine might be faster than processing the file using the CVP engine. For example, if the data set has indexes, the indexes are not used in order to optimize the WHERE expression if you use the CVP engine.

Note: The “CASNCHARMULTIPLIER= System Option” in SAS Cloud Analytic Services: User’s Guide increases the number of bytes when transcoding to UTF-8 in the CAS server.

Note: CVP creates a READONLY copy of the data. If you want to save a permanent copy of the data, you need to create a new data set.

Note: Data set and variable labels have a maximum of 256 bytes and variable names have a maximum of 32 characters. If a data set contains long labels with international characters, the transcoded labels might exceed the maximum, resulting in a transcoding error. You can shorten the long labels in the original data set to resolve the transcoding issue.

For more information and examples, see “CVPBYTES=, CVPPRINT=, CVPMULTIPLIER=, CVPPRINTWIDTH=, CVPVCHAR=, CVPPRINT=, CVPPRINTWIDTH= Statement” on page 731.

Avoiding Character Truncation Using the %COPY_TO_NEW_ENCODING Macro

If you write a CNTLOUT= data set using a different encoding than the session encoding, you might receive an error for truncating character variables. If this error occurs, write the CNTLOUT= data set using the current session encoding, and then
use the %COPY_TO_NEW_ENCODING macro to create the CNTLOUT= data set with the new encoding.

Suppose that you are using a data set that contains monetary values in Euros. Your session encoding is WLATIN1, and you specify UTF-8 encoding for the CNTLOUT= data set. In WLATIN1, the Label variable is predetermined to be 5 bytes long, with a value of €1234 (in hexadecimal, '803132334'x). When you attempt to store the variable in the CNTLOUT= data set with UTF-8 encoding, the length of that string must be 7 bytes (in hexadecimal, 'E282AC31323334'x), so it truncates. You can use the %COPY_TO_NEW_ENCODING macro to prevent this error.

For more information about the %COPY_TO_NEW_ENCODING macro, see “%COPY_TO_NEW_ENCODING Macro Function” on page 991.
Overview to Double-Byte Character Sets (DBCS)

Because East Asian languages have thousands of characters, double (two) bytes of information are needed to represent most characters.

Each East Asian language usually has more than one DBCS encoding system, due to nonstandardization among computer manufacturers. SAS processes the DBCS encoding information that is unique to each manufacturer for the major East Asian languages.

With the proper software extensions, you can use SAS for the following functions:

- display any of the major East Asian languages in the DBCS version of the SAS System
- import data from East Asian language computers and move the data from one application or operating environment to another (which might require SAS ACCESS or other SAS products)
- convert standard East Asian date and time notation to SAS date values, SAS time values, and SAS datetime values
- create data sets and various types of output (such as reports and graphs) that contain East Asian language characters.
East Asian Languages

East Asian languages include:

- Chinese, which is written in Simplified Chinese script, and is used in the People's Republic of China and Singapore
- Chinese, which is written in Traditional Chinese script, and is used in Hong Kong Special Administrative Region of the People's Republic of China (SAR), Macau SAR, and Taiwan
- Japanese
- Korean

Specifying DBCS

To specify DBCS, use the following SAS system options:

ENCODING
recognizes default character set encoding.

DBCS
recognizes DBCS characters.

You can set the session encoding by using the ENCODING= system option, the DBCS options, or the LOCALE= system option. The ENCODING= system option has priority. For more information, see “Setting the Encoding of a SAS Session” on page 21. Please see the “ENCODING System Option: UNIX, Windows, and z/OS” on page 694 for more information.

Requirements for Displaying DBCS Character Sets

In order to display data sets that contain DBCS characters, you must have the following resources:

- system support for multiple code pages
- DBCS fonts that correspond to the language that you intend to use

If you need to create a user-defined character for use with SAS software, your computer must support DBCS. These computers have a limited availability in the U.S. and Europe. These East Asian language computer systems use various methods of creating the characters. In one popular method, the user enters the phonetic pronunciation of the character, often using Latin characters. The computer
presents a menu of characters whose sounds are similar to the phonetic pronunciation and prompts the user to select one of them.

When You Can Use DBCS Features

After you have set up your SAS session to recognize a specific DBCS language and operating environment, you can work with your specified language in these general areas:

- the DATA step and batch-oriented procedures
- windowing and interactive capabilities
- cross-system connectivity and compatibility
- access to databases
- graphics

In a DATA step and in batch-oriented procedures, you can use DBCS wherever a text string within quotation marks is allowed. Variable values, variable labels, and data set labels can all be in DBCS. DBCS can also be used as input data and with range and label specifications in the FORMAT procedure. In WHERE expression processing, you can search for embedded DBCS text.

DBCS and SAS on a Mainframe

Another type of DBCS encoding exists on mainframe systems, which combine DBCS support with the 3270-style data stream. Each DBCS character string is surrounded by escape codes called shift out/shift in, or SO/SI. These codes originated from the need for the old-style printers to shift out from the EBCDIC character set, to the DBCS character set. The major manufacturers have different encodings for SO/SI; some manufacturers pad DBCS code with one byte of shift code information while others pad the DBCS code with two bytes of shift code information. These differences can cause problems in reading DBCS information about mainframes.

PCs, minicomputers, and workstations do not have SO/SI but have their own types of DBCS encodings that differ from manufacturer to manufacturer. SAS has several formats and informats that can read DBCS on SO/SI systems:

| Table 5.1 SAS Formats and Informats That Support DBCS on SO/SI Systems |
|--------------------------|--------------------------|--------------------------|
|Keyword | Language Element | Description |
| $KANJI | informat | Removes SO/SI from Japanese kanji DBCS |
| $KANJIX | informat | Adds SO/SI to Japanese kanji DBCS |
SAS Data Conversion between DBCS Encodings

Normally, DBCS data that is generated on one computer system is incompatible with data generated on another computer system. SAS has features that allow conversion from one DBCS source to another, as shown in the following table.

<table>
<thead>
<tr>
<th>Language Element</th>
<th>Type</th>
<th>Use</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCVT</td>
<td>function</td>
<td>Converts DBCS data from one operating environment to another</td>
<td>“KCVT Function” (p. 414)</td>
</tr>
<tr>
<td>CPORT</td>
<td>procedure</td>
<td>Moves files from one environment to another</td>
<td>Base SAS Procedures Guide</td>
</tr>
<tr>
<td>CIMPORT</td>
<td>procedure</td>
<td>Imports a transport file created by CPORT</td>
<td>Base SAS Procedures Guide</td>
</tr>
</tbody>
</table>

Avoiding Problems with Split DBCS Character Strings

- When working with DBCS characters, review your data to make sure that SAS recognizes the entire character string when data is imported or converted or used in a DATA step or a PROC step.
- On mainframe systems that use shift out/shift in escape codes, DBCS character strings can become truncated during conversion across operating environments.
- There is a possibility that DBCS character strings can be split when working with the PRINT, REPORT, TABULATE, and FREQ procedures. If undesirable splitting occurs, you might have to add spaces on either side of your DBCS character.
string to force the split to occur in a better place. The SPLIT= option can also be used with PROC REPORT and PROC PRINT to force string splitting in a better location.
# Time Zones

## Overview of SAS Time Zones

SAS must process local and international time and date values to accommodate international customers. This chapter explains time zones and how SAS processes the following:

- time zones
- Daylight Saving Time (DST)
- local time
- SAS datetime values

## Specifying Time Zones in SAS

### Definitions

**Daylight Saving Time (DST)**

Also referred to as *Summer Time, Daylight Saving Time (DST)*, it is a way of making better use of the daylight in the evenings by setting the clocks forward one hour during the longer days of summer and back again in the fall. DST begins in the northern hemisphere between March–April and ends between September–November. Standard time begins in the northern hemisphere between September–November and ends between March–April. DST begins in the southern hemisphere between September–November and ends between March–April. Standard time begins in the southern hemisphere between March–
April and ends between September–November. Many of the countries in the northern hemisphere observe DST. DST is not a standard for all countries.

Local time
Specify the local time when a time zone is not specified by the TIMEZONE= system option.

Time zone
An area of the earth’s surface in which an identical and standard time is used. It is usually referred to as the local time. Time zones have political and geographical boundaries and might be adjusted for the convenience of local populations. Some geographically large countries, such as India and China, use only one time zone, but other large countries, such as Russia and the United States, have more than one time zone.

Time zone ID
Specifies a region and area separated by a forward slash (/). Asia/Tokyo is a time zone ID. Time zone IDs are compatible with Java time zones. For a list of time zone IDs, see Appendix 3, “Time Zone IDs and Time Zone Names,” on page 953.

Time zone name
Specifies three or four characters that indicate a time zone. For a list of time zone names, see Appendix 3, “Time Zone IDs and Time Zone Names,” on page 953.

Time zone offset
Specifies the number of hours and minutes that a time zone is off from the Universal Coordinated Time (UTC) in the form +|-hh:mm or +|-hhmm.

User local time
Specify the local time for the time zone that is specified by the TIMEZONE= system option.

Universal Coordinated Time (UTC)
Specify the time at the zero meridian, near Greenwich, England. UTC is a datetime value that uses the ISO 8601 basic form yyyymmddThhmmss+|-hhmm or the ISO 8601 extended form yyyy-mm-ddThh:mm:ss+|-hh:mm.

About Time Zones in SAS
SAS datetime values are measured in seconds, beginning with 01Jan1960 00:00:00 local time. For example, the SAS datetime value for 01Jan1960 00:00:00 in England is 0. In Japan, the SAS datetime value for 01Jan1960 00:00:00 is also 0. There is a nine-hour difference between England and Japan. At the same point in time, the datetime in England and the datetime in Japan cannot be 0. In order to work with absolute time and datetime values, SAS supports the UTC date, time, and datetime values in time zones.

You can name a time zone either by specifying a time zone ID or a time zone name. Time zone IDs name a region and an area in the form region/area. An example of a time zone ID is America/New_York. A time zone name specifies the time zone. An example of a time zone name is EST for Eastern Standard Time. When a specific time zone changes the time for DST or summer time, a separate time zone name is available for that alternate time. PST is Pacific Standard Time. PDT is Pacific Daylight Time. EET is Eastern European Time. EEST is Eastern European Summer Time. When you specify a time zone ID, SAS determines the time using DST. You
do not need to determine whether the region/area uses alternate times. For a list of
time zone IDs and time zone names, see Appendix 3, “Time Zone IDs and Time
Zone Names,” on page 953.

When you specify a time zone, SAS shifts the datetime value according to the time
zone and stores the new value. The time-zone-specific timestamp is displayed to
the SAS log. The SAS timestamp constant enables you to specify a timestamp
using the ISO 8601 standard. Here is a timestamp using the SAS timestamp
constant:

tstamp='2013-05-17T09:15:30–05:00'dt;

When you specify a SAS timestamp constant, SAS converts the timestamp to a
local datetime value by using the current time zone and the time zone offset.

Time zone settings affect the TIME( ) function, the TODAY( ) function, the DATE( )
function, and the DATETIME( ) function. When a time zone is set, the date and
datetime values use the current time zone.

This example shows how the datetime values change according to the time zone.

```
options tz='America/New_York';
  data NY;
    NY_DT=datetime();
    format NY_DT datetime20.;
  run;
options tz='Asia/Shanghai';
  data SH;
    SH_DT=datetime();
    format SH_DT datetime20.;
  run;

options tz='America/New_York';
  proc contents data = NY nodetails;
  run;
options tz='Asia/Shanghai';
  proc contents data = NY nodetails;
  run;
```

These statements correspond to the lines that are numbered in the preceding DATA
and PROC steps.

1. Set the TZ option to America/New_York.
2. Assign the date and time to the variable NY_DT using the DATETIME function.
3. Modify the value of NY_DT using the DATETIME format.
4. Set the TZ option for the second DATA step to Asia/Shanghai.
5. Assign the date and time to the variable SH_DT using the DATETIME function.
6. Modify the value of SH_DT using the DATETIME format.
7. Set the TZ option for the first PROC CONTENTS step to America/New_York.
8. Display information for the data set NY. The TZ option is set to America/
    New_York.
9. Set the TZ option for the second PROC CONTENTS step to Asia/Shanghai.
10. Display information for the data set NY with the TZ option that is set to Asia/
    Shanghai. Notice the time zone shift in the Created and Last Modified rows. You
    are displaying information about the same data set but with different time zones.
These statements produce this output. The first output is from the data set using the America/New_York time zone. The second output is from the data set using the Asia/Shanghai time zone.

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>Observations</th>
<th>Member Type</th>
<th>Variables</th>
<th>Engine</th>
<th>Indexes</th>
<th>Created</th>
<th>Observation Length</th>
<th>Last Modified</th>
<th>Deleted Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK.NY</td>
<td></td>
<td>DATA</td>
<td>1</td>
<td>V9</td>
<td>0</td>
<td>27/02/2019 17:17:42</td>
<td>8</td>
<td>27/02/2019 17:17:42</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>Observations</th>
<th>Member Type</th>
<th>Variables</th>
<th>Engine</th>
<th>Indexes</th>
<th>Created</th>
<th>Observation Length</th>
<th>Last Modified</th>
<th>Deleted Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK.NY</td>
<td></td>
<td>DATA</td>
<td>1</td>
<td>V9</td>
<td>0</td>
<td>28/02/2019 06:17:42</td>
<td>8</td>
<td>28/02/2019 06:17:42</td>
<td>0</td>
</tr>
</tbody>
</table>

Tasks and SAS Language Elements for Time Zone Processing

Set a Time Zone

To set a time zone, use the TIMEZONE= system option:

```sas
options timezone='asia/tokyo';
```

Use time zone names or time zone IDs for the value of the option. Enclose the value in quotation marks. For time zone names and time zone IDs, see Appendix 3, "Time Zone IDs and Time Zone Names," on page 953. For more information, see "TIMEZONE= System Option" on page 709.

Determine a Time Zone ID Offset

You use the TZONEOFF( ) function to determine a time zone name or time zone ID offset:

- The TZONEOFF( ) function returns the time zone offset for the current time zone.
- The TZONEOFF('time-zone-id') function returns the time zone offset for the 'time-zone-id'.

This program returns the time zone offset for the current time zone (EST) and for Tokyo. Here is the output in the SAS log:
To find the difference between two time zones, you can use the ABS( ) function:

\[
\text{diff} = \text{abs}(\text{tzoneoff('america/new_york')} - \text{tzoneoff('asia/tokyo')})
\]

For more information, see “TZONEOFF Function” on page 498.

Determine a Time Zone ID or Time Zone Name

Use these functions to determine a time zone name or time zone ID:

- The TZONEID( ) function returns the current time zone ID.
- The TZONENAME( ) function returns the current time zone name based on a time zone ID and a SAS datetime value or based on a SAS datetime value only.

Here are some examples of using these functions. March 10 of 2013 is the first day of DST:

```sas
options timezone='America/Chicago';
data _null_
  tzid=tzoneid();
  put 'Current time zone is ' tzid;
  tzn=tzonename('america/los_angeles');
  put 'Time zone for Los Angeles: ' tzn;
  tznST=tzonename('america/los_angeles','10mar2013:01:00:00'dt);
  put 'Time zone for Los Angeles standard time: ' tznST;
  tznDT=tzonename('america/los_angeles','10mar2013:02:00:00'dt);
  put 'Time zone for Los Angeles daylight time: ' tznDT;
  tznSDT=tzonename('10mar2013:02:00:00'dt);
  put 'Time zone name for this SAS datetime: ' tznSDT;
run;
```

Here is the output:

```
Current time zone is AMERICA/CHICAGO
Time zone for Los Angeles: PST
Time zone for Los Angeles standard time: PST
Time zone for Los Angeles daylight time: PDT
Time zone name for this SAS datetime: CDT
```

For more information, see “TZONENAME Function” on page 497.

Convert Datetime Values between SAS and UTC

These functions convert SAS datetime values to UTC and UTC to SAS datetime values:

- The TONES2U( ) converts a SAS datetime value to a UTC datetime value.
The TZONEU2S( ) function converts a UTC datetime value to a SAS datetime value.

Here are some examples of using these functions:

```sas
options timezone='est';
data _null_;    
   put ' The time zone is EST';
   diff=abs(tzoneoff('america/new_york') - tzoneoff('europe/london'));    
   put ' New York-London difference: ' diff time.;
   diff=abs(tzoneoff('america/new_york') - tzoneoff('asia/tokyo'));    
   put ' New York-Tokyo difference: ' diff time.;
   put ' The SAS datetime is 2013-03-15T09:00+00:00 ';    
   put ' Change a SAS datetime to a UTC value ';    
   put ' The time zone offset +00:00 is for London ';    
   put ' Subtract the 5 hours for the EST time zone offset';    
   stu1=tzones2u('2013-03-15T09:15:00+00:00'dt);    
   put ' STU1 Using E8601DX: ' stu1 e8601dx.;    
   put ' 2013-03-15 9:15 AM in Tokyo is 2013-03-14 7:15 PM in New York';    
   put ' Subtract the 5 hours for the EST time zone offset';    
   stu2=tzones2u('2013-03-15T09:15:00+00:00'dt, 'Asia/Tokyo');    
   put ' STU2 Using E8601DX: ' stu2 e8601dx.;    
   put ' Change a UTC to a SAS datetime value. ';    
   put ' +00:00 is the time zone offset for London.';    
   put ' Subtract the 5 hours for the EST time zone offset';    
   uts1=tzoneu2s('2013-03-15T09:15:00+00:00'dt);    
   put ' UTS1 Using DATETIME: ' uts1 datetime.;    
   put ' 9:15:00+00:00 is 18:15:00 in Tokyo. ';    
   put ' Subtract the 5 hours for the EST time zone offset';    
   uts2=tzoneu2s('2013-03-15T09:15:00+00:00'dt, 'Asia/Tokyo');    
   put ' UTS2 Using DATETIME: ' uts2 datetime.;    
run;
```

Here is the output:
The time zone is EST
New York-London difference: 5:00:00
New York-Tokyo difference: 14:00:00
The SAS datetime is 2013-03-15T09:15:00+00:00

Change a SAS datetime to a UTC value

The time zone offset +00:00 is for London
Subtract the 5 hours for the EST time zone offset
STU1 Using E8601DX: 2013-03-15T04:15:00-05:00

2013-03-15 9:15 AM in Tokyo is 2013-03-14 7:15 PM in New York
Subtract the 5 hours for the EST time zone offset
STU2 Using E8601DX: 2013-03-14T14:15:00-05:00

Change a UTC to a SAS datetime value.

+00:00 is the time zone offset for London.
Subtract the 5 hours for the EST time zone offset
UTS1 Using DATETIME: 14MAR13:23:15:00

9:15:00+00:00 is 18:15:00 in Tokyo.
Subtract the 5 hours for the EST time zone offset
UTS2 Using DATETIME: 15MAR13:13:15:00

For more information, see “TZONES2U Function” on page 500.

Write ISO 8601 Time and Datetime Values Based on the Time Zone

These formats write ISO 8601 time and datetime values using basic and extended notation. The time, the UTC offset, or both are based on the value of the TIMEZONE= system option:

- The B8601TX. and E8601TX. formats convert a UTC time value to a user local time and write the time value with a time zone offset.
- The B8601DX. and E8601DX. formats convert a UTC datetime value to a user local time and write the datetime value with a time zone offset.
- The B8601LX. and E8601LX. formats write a local datetime value using the UTC offset for the user local time zone.

This program formats time and datetime values based on the time zone ID Australia/Sydney:

options timezone='Australia/Sydney';
data _null_;  
st='18:33:40't;  
sdt='2013-03-17T14:30:22+00:00'dt;  
put 'Time B8601TX:' st b8601tx.;  
put 'Time E8601TX:' st e8601tx.;  
put 'UTC B8601DX:' sdt b8601dx.;  
put 'UTC E8601DX:' sdt e8601dx.;  
put 'UTC B8601LX:' sdt b8601lx.;  
put 'UTC E8601LX:' sdt e8601lx.;  
run;

Here is the output:
Write SAS Datetime Values Using a Time Zone Offset

These formats write SAS datetime values using a UTC offset based on the value of the TIMEZONE= system option:

- The NLDATMZ. format writes a SAS datetime value using the form `ddmmmyyyy:hh:mm:ss +|-.hhmm`.
- The NLDATMTZ. format writes a SAS time value using the form `hh:mm:ss +|-.hhmm`.
- The NLDATMWZ. format writes a SAS datetime value as the day of the week, the month, the day, the year, and AM | PM in the form `day-of-week, month-name dd, yyyy AM|PM +hhmm`.

This program formats SAS time and datetime values using a time zone offset:

```sas
options timezone='Indian/Maldives';
data _null_;  
st='18:33:40't;  
sdt='2013-03-17T14:30:22+00:00'dt;  
put 'Time NLDATMTZ:' st nldatmtz.;  
put 'SAS datetime NLDATMZ:' sdt nldatmz.;  
put 'SAS datetime NLDATMWZ:' sdt nldatmwz.;  
run;
```

Here is the output:

```
Time NLDATMTZ:18:33:40 +0500
SAS datetime NLDATMZ:17Mar2013:19:30:22 +0500
SAS datetime NLDATMWZ:Sunday, March 17, 2013 07:30:22 PM +0500
```

For more information, see "NLDATMTZw. Format" on page 189, "NLDATMZw. Format" on page 203, and "NLDATMWZw. Format" on page 192.

Time Zone Example

This example shows the arrival time of a flight from Los Angeles to Tokyo:

```sas
/* Set the time zone */
options timezone='America/Los_Angeles';
data depart;
/* Set the departure time */
    depart='2013-05-17T09:15:00-08:00'dt;
    put 'Depart Los Angeles: ' depart nldatmtz.;
/* Set the flight time */
```
ftime='13:00't;
put 'Flight time=' ftime time.;
utc=depart+ftime;
put 'Arrive PST=' utc nldatmwz.;
put 'Arrive UTC=' utc nldatmwz.;
run;
/* Set the time zone for Tokyo */
options timezone='Asia/Tokyo';
data arrive;
  set depart;
  put 'Arrive in Tokyo ' utc nldatmwz.;
run;

Here is the output:

39 /* Set the time zone */
40 options timezone='America/Los_Angeles';
41 data depart;
42 /* Set the departure time */
43   depart='2013-05-17T09:15:00-08:00' dt;
44   put 'Depart Los Angeles: ' depart nldatmwz.;
45 /* Set the flight time */
46   ftime='13:00't;
47   put 'Flight time=' ftime time.;
48   utc=depart+ftime;
49   put 'Arrive PST=' utc nldatmwz.;
50   put 'Arrive UTC=' utc nldatmwz.;
51 run;

Depart Los Angeles: Friday, May 17, 2013 10:15:00 AM -0700
Flight time=13:00:00
Arrive PST=Friday, May 17, 2013 11:15:00 PM -0700
Arrive UTC=Friday, May 17, 2013 11:15:00 PM -0700
NOTE: The data set WORK.DEPART has 1 observations and 3 variables.
NOTE: DATA statement used (Total process time):
   real time 0.01 seconds
   cpu time 0.01 seconds

52 /* Set the time zone for Tokyo */
53 options timezone='Asia/Tokyo';
54 data arrive;
55   set depart;
56   put 'Arrive in Tokyo ' utc nldatmwz.;
57 run;

Arrive in Tokyo Friday, May 17, 2013 11:15:00 PM +0900
NOTE: There were 1 observations read from the data set WORK.DEPART.
NOTE: The data set WORK.ARRIVE has 1 observations and 3 variables.
NOTE: DATA statement used (Total process time):
   real time 0.01 seconds
   cpu time 0.01 seconds
Chapter 7

Dictionary of Autocall Macros for NLS
Dictionary of Autocall Macros for NLS

Autocall Macro Entries by Category

The following table provides brief descriptions of the SAS NLS autocall macros. For more detailed descriptions, see the NLS entry for each macro.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCS</td>
<td>%KLOWCASE Autocall Macro%QKLOWCAS Autocall Macro (p. 64)</td>
<td>Change uppercase characters to lowercase.</td>
</tr>
<tr>
<td></td>
<td>%KTRIM Autocall Macro %QKTRIM Autocall Macro (p. 64)</td>
<td>Trim trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%KVERIFY Autocall Macro (p. 65)</td>
<td>Returns the position of the first character unique to an expression.</td>
</tr>
</tbody>
</table>
Dictionary

%KLOWCASE Autocall Macro

%QKLOWCAS Autocall Macro

Change uppercase characters to lowercase.
Category: DBCS
Requirement: MAUTOSOURCE system option

Syntax

%KLOWCASE (text | text expression)
%QKLOWCAS (text | text expression)

Details

The %KLOWCASE and %QKLOWCAS macros change uppercase alphabetic characters to their lowercase equivalents. If the argument might contain a special character or mnemonic operator, listed below, use %QKLOWCAS.

%KLOWCASE returns a result without quotation marks, even if the argument has quotation marks. %QKLOWCAS produces a result with the following special characters and mnemonic operators masked so that the macro processor interprets them as text instead of as elements of the macro language:

& % ' " ( ) + - / < > = ¬ ^ ~ ; , blank AND OR NOT EQ NE LE LT GE GT IN

Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel.

%KTRIM Autocall Macro

%QKTRIM Autocall Macro

Trim trailing blanks.
Category: DBCS
Requirement: MAUTOSOURCE system option

Syntax

%KTRIM (text | text expression)
%QKTRIM (text | text expression)

Details

The KTRIM macro and the QKTRIM macro trim trailing blanks. If the argument contains a special character or mnemonic operator, listed below, use %QKTRIM.

QKTRIM produces a result with the following special characters and mnemonic operators masked so that the macro processor interprets them as text instead of as elements of the macro language:

& % ' " ( ) + - / > = ¬ ? ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN

Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel.

%KVERIFY Autocall Macro

Returns the position of the first character unique to an expression.

Category: DBCS
Requirement: MAUTOSOURCE system option

Syntax

%KVERIFY (source, excerpt)

Syntax

source
is text or a text expression that you want to examine for characters that do not exist in excerpt.

excerpt
is text or a text expression that defines the set of characters that %KVERIFY uses to examine source.

Details

%KVERIFY returns the position of the first character in source that is not also present in excerpt. If all characters in source are present in excerpt, %KVERIFY returns 0.

Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you
want to find out if the library is a site-specific version, see your on-site SAS support personnel.
Dictionary of Data Set Options for NLS

Data Set Options by Category

NLS affects the data set control category of options for selected data set options. The following table provides brief descriptions of the data set options. For more detailed descriptions, see the dictionary entry for each data set option:

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Control</td>
<td>ENCODING= Data Set Option (p. 69)</td>
<td>Overrides the encoding to use for reading or writing a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>OUTREP= Data Set Option (p. 72)</td>
<td>Specifies the data representation for the output SAS data set.</td>
</tr>
</tbody>
</table>

Dictionary

ENCODING= Data Set Option

Overrides the encoding to use for reading or writing a SAS data set.

Valid in: DATA step and PROC steps

Category: Data Set Control
Syntax

ENCODING= ANY | ASCIIANY | SESSION | EBCDICANY | encoding-value

Syntax Description

ANY
specifies that no transcoding occurs.

Note ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

ASCIIANY
specifies that no transcoding occurs when the mixed encodings are ASCII encodings.

SESSION
specifies the locale of the current Compute Server session.

The session value is supported in the OVERRIDE= option in the COPY procedure or COPY statement. For more information, see OVERRIDE=.

EBCDICANY
specifies that no transcoding occurs when the mixed encodings are EBCDIC encodings.

encoding-value
specifies an encoding value.

See Chapter 3, “Encoding for NLS,” on page 9

Details

The value for ENCODING= indicates that the SAS data set has a different encoding from the current session encoding. When you read data from a data set, SAS transcodes the data from the specified encoding to the session encoding. When you write data to a data set, SAS transcodes the data from the session encoding to the specified encoding.

Input Processing

By default, encoding for input processing is determined as follows:

- If the session encoding and the encoding that is specified in the file are different, SAS transcodes the data to the session encoding.
- If a file has no encoding specified, but the file’s data representation is different from the encoding of the current session, then SAS transcodes the data to the current session.

Output Processing

By default, encoding for output processing is determined as follows:

- Data is written to a file using the encoding of the current session, except when a different output representation is specified using the OUTREP= data set option, the OUTENCODING= option in the LIBNAME statement, or the ENCODING= data set option.
- If a new file replaces an existing file, then the new file inherits the encoding of the existing file.
If an existing file is replaced by a new file that was created under a different operating environment or that has no encoding specified, the new file uses the encoding of the current session.

**Note:** Character metadata and data output appears garbled if you specify a different encoding from where the data set was created. In this example, the data set to be printed is internally encoded as ASCII. However, the data set option specifies an EBCDIC encoding. SAS attempts to transcode the data from EBCDIC to ASCII, but the data is already in ASCII. The result is garbled data.

```sas
data a;
x=1;
abc='abc';
run;
proc print data=a (encoding="ebcdic");
run;
```

**Note:** The following values for **ENCODING=** are invalid:

- UCS2
- UCS4
- UTF16
- UTF32

**Comparisons**

- Session encoding is specified using the **ENCODING=** system option or the **LOCALE=** system option, with each operating environment having a default encoding.
- You can specify encoding for a SAS library by using the **LIBNAME** statement's **INENCODING=** option (for input files) and the **OUTENCODING=** option (for output files). If both the **LIBNAME** statement option and the **ENCODING=** data set option are specified, SAS uses the data set option.

**Examples:**

**Example 1: Creating a SAS Data Set with Mixed Encodings and with Transcoding Suppressed**

By specifying the data set option **ENCODING=ANY**, you can create a SAS data set that contains mixed encodings, and suppress transcoding for either input or output processing.

In this example, the new data set **MYFILES.MIXED** contains some data that uses the Latin1 encoding, and some data that uses the Latin2 encoding. When the data set is processed, no transcoding occurs. For example, the correct Latin1 characters in a Latin1 session encoding and correct Latin2 characters in a Latin2 session encoding are displayed.

```sas
libname myfiles 'SAS data-library';
data myfiles.mixed (encoding=any);`
Example 2: Creating a SAS Data Set with a Particular Encoding

For output processing, you can override the current session encoding. This action might be necessary, for example, if the normal access to the file uses a different session encoding.

For example, if the current session encoding is WLatin1, you can specify ENCODING=WLatin2 in order to create the data set that uses the encoding WLatin2. The following statements tell SAS to write the data to the new data set using the WLatin2 encoding instead of the session encoding. The encoding is also specified in the descriptor portion of the file.

```sas
libname myfiles 'SAS data-library';
data myfiles.difencoding (encoding=wlatin2);
run;
```

Example 3: Overriding Encoding for Input Processing

For input processing, you can override the encoding that is specified in the file, and specify a different encoding.

For this example, the current session encoding is EBCDIC-870, but the file has the encoding value EBCDIC-1047 in the descriptor information. By specifying ENCODING=EBCDIC-870, SAS does not transcode the data, but instead displays the data using EBCDIC-870 encoding.

```sas
proc print data=myfiles.mixed (encoding=ebcidic870);
run;
```

See Also

- Conceptual discussion in Chapter 3, “Encoding for NLS,” on page 9

Options in Statements and Commands:

- "ENCODING= Option Statement" on page 742
- "INENCODING=, OUTENCODING= Options Statements" on page 745

System Options:

- "ENCODING System Option: UNIX, Windows, and z/OS" on page 694
- "LOCALE System Option" on page 698

OUTREP= Data Set Option

Specifies the data representation for the output SAS data set.
Valid in: DATA step and PROC steps
Category: Data Set Control
See: “OUTREP= Data Set Option” in SAS Data Set Options: Reference.

Syntax

OUTREP= format
PART 4

Formats for NLS

Chapter 9
Overview to NLS Formats ............................................. 77

Chapter 10
Dictionary of Formats for NLS ........................................ 89
International Date and Datetime Formats

SAS supports international formats that are equivalent to some of the most commonly used English-language date formats. In each case the format works like the corresponding English-language format. Only the maximum, minimum, and default widths are different.

Table 9.1 International Date and Datetime Formats

<table>
<thead>
<tr>
<th>English Format</th>
<th>International Format</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
</tbody>
</table>
Currency Representation

Overview to Currency

Currency is the medium of exchange, which is specific to a country. SAS provides formats and informats for reading and writing currency.

U.S. Dollars

The DOLLARw.d formats and informats were first introduced to read and write American currency. DOLLARw.d

- uses the dollar sign ($) currency symbol to precede U.S. currency
- uses a comma (,) as the thousands separator and a dot (.) as the decimal separator.

Example:

$12,345.00

DOLLARXw.d also writes currency with a leading dollar sign ($), but uses a dot (.) as the thousands separator and a comma (,) as the decimal separator. The reversal of the dot and comma for currency formatting is a convention used in many European countries.

Example:

$12.345,00

Limitations of the DOLLAR formats and informats are:

- the lack of support for all currency symbols
- the reversal of the dot and comma for currency formatting is not used by all European countries.
- the appearance of the currency symbol varies by computer (an EBCDIC-based computer and an ASCII-based computer render characters differently).

Euros

The EUROw.d formats and informats were introduced to support the euro currency that was established by the European Monetary Union (EMU), which was formed in 1999. EUROw.d

Example:

options locale=English_UnitedKingdom;
data _null_;
Customized Currency Representations

To create a customized currency representation, you can use the FORMAT procedure. The following example shows the creation of unique formats for the Australian dollar, the Swiss franc, and the British pound. For details about the FORMAT procedure, see Base SAS Procedures Guide.

Example Code 9.1 SAS Code That Customizes Currency Representations

```sas
proc format;
   picture aud low-<0='0,000,000,009.00'
               (prefix='-AU$' mult=100)
          0–high='0,000,00,009.00 '
               (prefix='AU$' mult=100);
   picture sfr low-<0='0,000,000,009.00'
                 (prefix='-SFr.' mult=100)
          0–high='0,000,00,009.00 '
                 (prefix='-SFr.' mult=100);
   picture bpd low-<0='0,000,000,009.00'
                 (prefix='-BPd.' mult=100)
          0–high='0,000,00,009.00 '
                 (prefix='BPd.' mult=100);
run;
```

```sas
data currency;
   input aud sfr bpd 12.2;
   datalines;
   12345 12345 12345
     0 0 0
   -12345 -12345 -12345
;```

```sas
proc print data=currency noobs;
   var aud sfr bpd;
   format aud aud. sfr sfr. bpd bpd.;
   title 'Unique Currency Formats';
   run;
```
Customizing currency representations offers flexibility, but requires a programming solution.

Localized National and International Currency Representations

The NLMNYw.d and NLMNYIw.d formats and informats were introduced to represent localized currency in two forms:

Localized national currency representation reflects the customs and conventions of the locale. National formats are specified using the NLMNYw.d formats and informats. You must also use the LOCALE= option to specify the locale when using the NLMNYw.d formats and informats.

Example:
```
options locale=english_UnitedStates;
data _null_;  
x=12345;  
   put x nlmny15.2;  
run;
```

Output:
```
$12,345.00
```

Selected national currency representations follow:

Table 9.2  Localized National Currency Representations

<table>
<thead>
<tr>
<th>LOCALE=</th>
<th>Currency</th>
<th>National Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English_UnitedStates</td>
<td>U.S. dollars</td>
<td>$12,345.00</td>
</tr>
<tr>
<td>French_Canada</td>
<td>Canadian dollars</td>
<td>12 345,00 $</td>
</tr>
<tr>
<td>French_France</td>
<td>Euros</td>
<td>12 345,00 €</td>
</tr>
<tr>
<td>French_Switzerland</td>
<td>Swiss francs</td>
<td>SFr. 12‘345.00</td>
</tr>
<tr>
<td>German_Germany</td>
<td>Euros</td>
<td>12.345,00 €</td>
</tr>
<tr>
<td>German_Luxembourg</td>
<td>Euros</td>
<td>12.345,00 €</td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>Euros</td>
<td>12.345,00 €</td>
</tr>
</tbody>
</table>
The localized renderings show the native customs for representing currency. For example, although these selected EMU countries might use the same euro currency, their depiction of the currency varies. Whereas French_France uses no thousands separator but uses a comma as a decimal separator, German_Germany and Spanish_Spain use a dot as a thousands separator and a comma as a decimal separator.

Localized International currency representation conforms to ISO standard 4217. International forms are specified using the NLMNYlw.d formats and informats. International forms are commonly used to show a comparison of world currencies. Examples are airline ticket, trade, and stock market pricing. You must also use the LOCALE= option to specify the locale when using the NLMNYlw.d formats and informats. The letter “I,” which signifies “International,” is appended to the format and informat names.

Example:

```plaintext
options locale=english_UnitedStates;
data _null_;  
x=12345; put x nlmnyi15.2;
run;
```

Output:

```
USD12,345.00
```

Selected international currency representations follow:

**Table 9.3  International Currency Representations by Locale (ISO standard 4217)**

<table>
<thead>
<tr>
<th>LOCALE=</th>
<th>Currency</th>
<th>International Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English_UnitedStates</td>
<td>U.S. dollars</td>
<td>USD12,345.00</td>
</tr>
<tr>
<td>French_Canada</td>
<td>Canadian dollars</td>
<td>12 345.00 CAD</td>
</tr>
<tr>
<td>French_France</td>
<td>Euros</td>
<td>12.345,00 EUR</td>
</tr>
<tr>
<td>French_Luxembourg</td>
<td>Euros</td>
<td>12.345,00 EUR</td>
</tr>
<tr>
<td>German_Germany</td>
<td>Euros</td>
<td>12.345,00 EUR</td>
</tr>
<tr>
<td>German_Switzerland</td>
<td>Swiss francs</td>
<td>CHF12,345.00</td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>Euros</td>
<td>12.345,00 EUR</td>
</tr>
<tr>
<td>Spanish_Venezuela</td>
<td>Venezuelan bolivars</td>
<td>VEF12.345.00</td>
</tr>
</tbody>
</table>

The international renderings also reflect native customs for representing currency. For example, not every locale uses a comma as the thousands separator.
separator and a dot as the decimal separator. When used, the ISO currency code placement varies by locale. Whereas the EMU countries put the currency code after the currency, English_UnitedStates, German_Switzerland, and Spanish_Venezuela precede the currency with the ISO code.

For a complete list of the ISO standard 4217 currency codes, see http://www.xe.com/iso4217.php.

A primary limitation of using localized national and international currency representations is their dependence on a value for the LOCALE= system option.

Unique National and International Monetary Representations

The NLMNY|ISOw.d and NLMNY|ISOw.d formats and informats were introduced to uniquely represent each currency without having to also use the LOCALE= option. Each currency is specified by a unique ISO standard 4217 currency code.

NLMNY and NLMNYI formats return results that display the number that is found in the data with a currency symbol that matches the current SAS LOCALE option. SAS does not convert the value from one currency to another. If the LOCALE changes, the same number is displayed with a different currency symbol. A changed currency symbol does not guarantee that the value has changed.

Unique national monetary representation is specified by the unique ISO currency code. National formats are specified using the NLMNL|ISOw.d formats and informats. In the following example, USD is the ISO currency code for American dollars.

Note: When using the NLMNL|ISOw.d formats and informats, the LOCALE= option is optional.

Example:

data _null_
  put x nlmnlusd15.2;
run;

Output:
US$12,345.00

Selected unique national currency representations follow:

In this example, locale is specified as fr_FR.

data _null_
  x=12345;
  put x nlmnleur15.2;
run;

Output:
€12,345.00
Table 9.4  Unique Currency Representations by ISO Currency Code

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Currency</th>
<th>National Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD</td>
<td>U.S. dollars</td>
<td>US$12,345.00</td>
</tr>
<tr>
<td>CAD</td>
<td>Canadian dollars</td>
<td>CA$12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>CHF</td>
<td>Swiss francs</td>
<td>SFr.12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>VEB</td>
<td>Venezuelan bolivars</td>
<td>BsF.12,345.00</td>
</tr>
</tbody>
</table>

A currency symbol or a currency code precedes most currencies. Also, a comma is used as the thousands separator and a dot as the decimal separator. If the currency symbol of the local currency is not supported in the current SAS session encoding, the NLMNLxxxw.d format, formats the value with the 3-letter ISO currency code.

Unique international monetary representation is specified by the unique ISO currency code. International formats are specified using the NLMNIISOw.d formats and informats. International forms are commonly used to show a comparison of world currencies (for example, airline ticket, trade, and stock market pricing). The letter “I”, which signifies “International”, is appended to the format and informat names. In the following example, USD is the ISO currency code for American dollars.

Note: When using the NLMNIISOw.d formats and informats, do not use the LOCALE= option to specify the locale.

    data _null_; 
      x=12345; 
      put x nlmniusd15.2; 
    run; 
    Output: 
      USD12,345.00

Selected international currency representations follow:
### Table 9.5  International Currency Representations by ISO Currency Code

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Currency</th>
<th>International Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD</td>
<td>U.S. dollars</td>
<td>USD12,345.00</td>
</tr>
<tr>
<td>CAD</td>
<td>Canadian dollars</td>
<td>CAD12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>CHF</td>
<td>Swiss francs</td>
<td>CHF12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>VEB</td>
<td>Venezuelan bolivars</td>
<td>Not found</td>
</tr>
</tbody>
</table>

The international renderings precede the currency with the appropriate ISO code. Also used are a comma as the thousands separator and a dot as the decimal separator.

---

**Example: Representing Currency in National and International Formats**

This SAS program uses the exchange rates for selected Asia-Pacific countries against the U.S. dollar. In the output, each country’s currency is represented using a national format and an international format.

**Example Code 9.2  SAS Code That Formats National and International Currency Formats**

```sas
data curr;
input ex_date mmddyy. usd aud hkd jpy sgd 12.2;
datalines;
061704 1.00000 1.45349 7.79930 110.110 1.71900;
proc print data=curr noobs label;
var ex_date usd aud hkd jpy sgd;
format ex_date mmddyy. usd nlmnlusd15.2 aud nlmnlaud15.2 hkd
nlmlhkd15.2 jpy nlmnljpy15.2 sgd nlmnlsgd15.2;
label ex_date='Date' usd='US' aud='Australia' hkd='Hong Kong'
jpy='Japan' sgd='Singapore';
title 'Exchange Rates for Selected Asian-Pacific Countries
(Localized Currency Codes)';
proc print data=curr noobs label;
var ex_date usd aud hkd jpy sgd;
```
format ex_date mmdyy. usd nlmiusd15.2 aud nlmiaud15.2 hkd nlmihkd15.2 jpy nlmijpy15.2 sgd nlmisgd15.2; 3
label ex_date='Date' usd='US' aud='Australia' hkd='Hong Kong' jpy='Japan' sgd='Singapore';
title 'Exchange Rates for Selected Asian-Pacific Countries
(International Currency Codes)';
run;

1 These exchange rates, which were effective June 17, 2004, are specified as
data in the SAS program.

2 These NLMNL/ISO formats are applied to each of the numeric data items that are
specified in the INPUT statement. These formats show currencies in the
appropriate national formats.

3 These NLMNI/ISO formats are applied to each of the numeric data items that are
specified in the INPUT statement. These formats show currencies in the
appropriate international formats.

Figure 9.1 National and International Format Output

Exceptions for Date and Time Default Widths

The length of the month-name and the week-name in some locale and encoding
combinations might exceed the default width in some formats. Do not use the
default width for these combinations. Refer to the following tables for the affected
locales, encodings, and formats. Use the recommended widths for accurate output.

The following table lists the encoding and locale combinations where the localized
output might exceed the default width.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Locale</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLATIN1</td>
<td>FR_FR, IT_IT, DE_DE, ES_ES,</td>
</tr>
<tr>
<td></td>
<td>EN_US, EN_GB</td>
</tr>
<tr>
<td>WLATIN2</td>
<td>CS_CS, HU_HU, PL.PL</td>
</tr>
<tr>
<td>WCYRILLIC</td>
<td>RU_RU</td>
</tr>
<tr>
<td>EUC_CN</td>
<td>EN_US &amp; ZH_CN</td>
</tr>
<tr>
<td>SHIFT-JIS</td>
<td>EN_US JA_JP4-6</td>
</tr>
</tbody>
</table>
The following table lists the recommended widths for encoding and locale combinations where the localized output might exceed the default width. You might need to use the recommended width in the specified formats.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Locale</th>
<th>Format</th>
<th>Default Width</th>
<th>Recommended Width (&gt;=)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUC-KR</td>
<td>EN_US KO_KR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIG5</td>
<td>EN_US ZT_TW ZT_HK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTF-8</td>
<td>all listed locales</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Locale</th>
<th>Format</th>
<th>Default Width</th>
<th>Recommended Width (&gt;=)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLATIN1</td>
<td>EN_GB</td>
<td>nldatmw</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>WLATIN2</td>
<td>CS_CS</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>EUC-CN</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>ZH_CN</td>
<td>nldatmw</td>
<td>30</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>SHIFT-JIS</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>JA_JP</td>
<td>nldatmw</td>
<td>30</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>EUC-KR</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>KO_KR</td>
<td>nldatmw</td>
<td>30</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>BIG5</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>ZT_HK</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>ZT_TW</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>UTF-8</td>
<td>CS_CS</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>EN_GB</td>
<td>nldatmw</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td>Encoding</td>
<td>Locale</td>
<td>Format</td>
<td>Default Width</td>
<td>Recommended Width (&gt;=)</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>JA_JP</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>KO_KR</td>
<td>nldatew</td>
<td>20</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>RU_RU</td>
<td>nldatew</td>
<td>20</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>ZH_CN</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>ZT_HK</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>ZT_TW</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>
## Categories of NLS Formats

<table>
<thead>
<tr>
<th>Dictionary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B8601DXw. Format</td>
<td>116</td>
</tr>
<tr>
<td>B8601LXw. Format</td>
<td>117</td>
</tr>
<tr>
<td>B8601TXw. Format</td>
<td>119</td>
</tr>
<tr>
<td>BESTDOTXw. Format</td>
<td>120</td>
</tr>
<tr>
<td>$BIDIw. Format</td>
<td>122</td>
</tr>
<tr>
<td>$CPTDWWw. Format</td>
<td>123</td>
</tr>
<tr>
<td>$CPTWDww. Format</td>
<td>124</td>
</tr>
<tr>
<td>DTWEEKVw. Format</td>
<td>125</td>
</tr>
<tr>
<td>$EBCDICww. Format</td>
<td>126</td>
</tr>
<tr>
<td>E8601DXw. Format</td>
<td>128</td>
</tr>
<tr>
<td>E8601LXw. Format</td>
<td>129</td>
</tr>
<tr>
<td>E8601TXw. Format</td>
<td>131</td>
</tr>
<tr>
<td>EUROw.d Format</td>
<td>132</td>
</tr>
<tr>
<td>EUROXw.d Format</td>
<td>135</td>
</tr>
<tr>
<td>HDATEww. Format</td>
<td>137</td>
</tr>
<tr>
<td>HEBDATEww. Format</td>
<td>139</td>
</tr>
<tr>
<td>JNENGOwwd Format</td>
<td>140</td>
</tr>
<tr>
<td>JNENGOTww. Format</td>
<td>141</td>
</tr>
<tr>
<td>JNENGOTww. Format</td>
<td>142</td>
</tr>
<tr>
<td>JNENGOww.d Format</td>
<td>143</td>
</tr>
<tr>
<td>$KANJww. Format</td>
<td>144</td>
</tr>
<tr>
<td>$KANJXww. Format</td>
<td>145</td>
</tr>
<tr>
<td>$LOGVSww. Format</td>
<td>146</td>
</tr>
<tr>
<td>$LOGVSww. Format</td>
<td>148</td>
</tr>
<tr>
<td>MINGUOww. Format</td>
<td>149</td>
</tr>
<tr>
<td>NENGOww. Format</td>
<td>151</td>
</tr>
<tr>
<td>NLBESTww. Format</td>
<td>152</td>
</tr>
<tr>
<td>NLDATEmww. Format</td>
<td>154</td>
</tr>
<tr>
<td>NLDATELww. Format</td>
<td>155</td>
</tr>
<tr>
<td>NLDATEmww. Format</td>
<td>156</td>
</tr>
<tr>
<td>NLDATEmDww. Format</td>
<td>157</td>
</tr>
<tr>
<td>NLDATEmDLww. Format</td>
<td>158</td>
</tr>
<tr>
<td>NLDATEmDMww. Format</td>
<td>159</td>
</tr>
<tr>
<td>NLDATEmDSSw. Format</td>
<td>160</td>
</tr>
<tr>
<td>NLDATEmNww. Format</td>
<td>161</td>
</tr>
<tr>
<td>NLDATESww. Format</td>
<td>162</td>
</tr>
<tr>
<td>NLDATEmWw. Format</td>
<td>163</td>
</tr>
<tr>
<td>NLDATENww. Format</td>
<td>165</td>
</tr>
<tr>
<td>NLDATEMYww. Format</td>
<td>166</td>
</tr>
<tr>
<td>NLDATEMYMLww. Format</td>
<td>167</td>
</tr>
<tr>
<td>Format</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>NLMNJPYw.d</td>
<td>222</td>
</tr>
<tr>
<td>NLMNIKRWw.d</td>
<td>223</td>
</tr>
<tr>
<td>NLMNILTLw.d</td>
<td>224</td>
</tr>
<tr>
<td>NLMNILVLw.d</td>
<td>225</td>
</tr>
<tr>
<td>NLMNIMOPw.d</td>
<td>226</td>
</tr>
<tr>
<td>NLMNIMXNw.d</td>
<td>227</td>
</tr>
<tr>
<td>NLMNIMYRw.d</td>
<td>228</td>
</tr>
<tr>
<td>NLMNINOKw.d</td>
<td>229</td>
</tr>
<tr>
<td>NLMNINZDw.d</td>
<td>230</td>
</tr>
<tr>
<td>NLMNIPLNW.d</td>
<td>231</td>
</tr>
<tr>
<td>NLMNIRUBw.d</td>
<td>232</td>
</tr>
<tr>
<td>NLMNISEKw.d</td>
<td>233</td>
</tr>
<tr>
<td>NLMNISGDw.d</td>
<td>234</td>
</tr>
<tr>
<td>NLMNITHBw.d</td>
<td>235</td>
</tr>
<tr>
<td>NLMNITRYw.d</td>
<td>236</td>
</tr>
<tr>
<td>NLMNITWDw.d</td>
<td>237</td>
</tr>
<tr>
<td>NLMNIUSDw.d</td>
<td>238</td>
</tr>
<tr>
<td>NLMNIZARw.d</td>
<td>239</td>
</tr>
<tr>
<td>NLMNLAEDx.d</td>
<td>240</td>
</tr>
<tr>
<td>NLMNLAUDw.d</td>
<td>241</td>
</tr>
<tr>
<td>NLMNLBGNw.d</td>
<td>242</td>
</tr>
<tr>
<td>NLMNLBRLw.d</td>
<td>243</td>
</tr>
<tr>
<td>NLMNLCADw.d</td>
<td>244</td>
</tr>
<tr>
<td>NLMNLCHFw.d</td>
<td>245</td>
</tr>
<tr>
<td>NLMNLCNYw.d</td>
<td>246</td>
</tr>
<tr>
<td>NLMNLCZKw.d</td>
<td>247</td>
</tr>
<tr>
<td>NLMNLDKKw.d</td>
<td>248</td>
</tr>
<tr>
<td>NLMNLEEKw.d</td>
<td>249</td>
</tr>
<tr>
<td>NLMNLEGPw.d</td>
<td>250</td>
</tr>
<tr>
<td>NLMNLLEURw.d</td>
<td>251</td>
</tr>
<tr>
<td>NLMNLGBPw.d</td>
<td>252</td>
</tr>
<tr>
<td>NLMNLHKDw.d</td>
<td>253</td>
</tr>
<tr>
<td>NLMNLHRKw.d</td>
<td>254</td>
</tr>
<tr>
<td>NLMNLHUFw.d</td>
<td>255</td>
</tr>
<tr>
<td>NLMNLIDRw.d</td>
<td>256</td>
</tr>
<tr>
<td>NLMNLILSw.d</td>
<td>257</td>
</tr>
<tr>
<td>NLMNLRINrw.d</td>
<td>258</td>
</tr>
<tr>
<td>NLMNLJPYw.d</td>
<td>259</td>
</tr>
<tr>
<td>NLMNLKRWw.d</td>
<td>260</td>
</tr>
<tr>
<td>NLMNLLTLw.d</td>
<td>261</td>
</tr>
<tr>
<td>NLMNLLLVw.d</td>
<td>262</td>
</tr>
<tr>
<td>NLMNLMOPw.d</td>
<td>263</td>
</tr>
<tr>
<td>NLMNLMXNw.d</td>
<td>264</td>
</tr>
<tr>
<td>NLMNMYRw.d</td>
<td>265</td>
</tr>
<tr>
<td>NLMNLOKw.d</td>
<td>266</td>
</tr>
<tr>
<td>NLMNINZDw.d</td>
<td>267</td>
</tr>
<tr>
<td>NLMNLPLNW.d</td>
<td>268</td>
</tr>
<tr>
<td>NLMNLRUBw.d</td>
<td>269</td>
</tr>
<tr>
<td>NLMNISEkw.d</td>
<td>270</td>
</tr>
<tr>
<td>NLMNLSGDw.d</td>
<td>271</td>
</tr>
<tr>
<td>NLMNLTTHBw.d</td>
<td>272</td>
</tr>
<tr>
<td>NLMNLRTRYw.d</td>
<td>273</td>
</tr>
<tr>
<td>NLMNLTWDw.d</td>
<td>274</td>
</tr>
<tr>
<td>NLMNUSDw.d</td>
<td>275</td>
</tr>
</tbody>
</table>
Chapter 10 / Dictionary of Formats for NLS

NLMNLZARw.d Format .................................................. 276
NLMNYw.d Format .................................................. 277
NLMNYlw.d Format .................................................. 279
NLNUMw.d Format .................................................. 281
NLNUMlw.d Format .................................................. 282
NLPCTw.d Format .................................................. 283
NLPCtlw.d Format .................................................. 285
NLPCTNw.d Format .................................................. 286
NLPCTPw.d Format .................................................. 287
NLVALUEw.d Format .................................................. 288
NLSTRMONw.d Format .................................................. 290
NLSTRQTRw.d Format .................................................. 291
NLSTRWKw.d Format .................................................. 292
NLTIMAPw. Format .................................................. 294
NLTIMEw. Format .................................................. 295
NLTIMElw. Format .................................................. 296
NLTIMEEMw. Format .................................................. 298
NLTIMESw. Format .................................................. 299
$UCS2Bw. Format .................................................. 300
$UCS2BEw. Format .................................................. 301
$UCS2Lw. Format .................................................. 303
$UCS2LEw. Format .................................................. 304
$UCS2Xw. Format .................................................. 305
$UCS2XEw. Format .................................................. 307
$UCS4Bw. Format .................................................. 308
$UCS4BEw. Format .................................................. 309
$UCS4Lw. Format .................................................. 311
$UCS4LEw. Format .................................................. 312
$UCS4Xw. Format .................................................. 313
$UCS4XEw. Format .................................................. 315
$UESCw. Format .................................................. 316
$UESCEw. Format .................................................. 317
$UNCRw. Format .................................................. 319
$UNCREw. Format .................................................. 320
$UPARENWw. Format .................................................. 321
$UPARENEw. Format .................................................. 322
$UTF8Xw. Format .................................................. 323
$UTF8XEw. Format .................................................. 324
$VSLOGw. Format .................................................. 325
$VSLOGRw. Format .................................................. 327
WEEKUw. Format .................................................. 328
WEEKVw. Format .................................................. 330
WEEKWw. Format .................................................. 331
YENw.d Format .................................................. 333
YYWEEKUw. Format .................................................. 334
YYWEEKVw. Format .................................................. 336
YYWEEKWw. Format .................................................. 338
Categories of NLS Formats

Some formats run in SAS only, and some formats run in SAS and on the CAS engine. If CAS is specified for the format category, then the format runs in SAS and on the CAS server. If CAS is not specified for the format category, then the format runs in SAS only. For example, the NLSDATE format runs in SAS and on the CAS server, so CAS is specified as a category. The $UNCR format runs on SAS only, so CAS is not specified as a category.

Here are categories related to NLS issues:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI Text Handling</td>
<td>Instructs SAS to write bidirectional data values from data variables.</td>
</tr>
<tr>
<td>CAS</td>
<td>Instructs SAS that these formats run on the CAS server.</td>
</tr>
<tr>
<td>Character</td>
<td>Instructs SAS to write character data values from character variables.</td>
</tr>
<tr>
<td>Currency Conversion</td>
<td>Instructs SAS to convert an amount from one currency to another currency.</td>
</tr>
<tr>
<td>Date</td>
<td>Instructs SAS to write data values from variables that represent dates.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>Instructs SAS to write data values from variables that represent dates, times, and datetimes.</td>
</tr>
<tr>
<td>DBCS</td>
<td>Instructs SAS to translate double-byte-character sets that are used in Asian languages.</td>
</tr>
<tr>
<td>Hebrew Text Handling</td>
<td>Instructs SAS to read Hebrew data from data variables.</td>
</tr>
<tr>
<td>International Date and Time</td>
<td>Instructs SAS to write data values from variables that represent dates, times, and datetimes.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Instructs SAS to write numeric data values from numeric variables.</td>
</tr>
<tr>
<td>Time</td>
<td>Instructs SAS to write data values from variables that represent times.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>BIDI Text Handling</td>
<td>$BIDIw. Format (p. 122)</td>
</tr>
<tr>
<td></td>
<td>$LOGVSw. Format (p. 146)</td>
</tr>
<tr>
<td></td>
<td>$LOGVSRw. Format (p. 148)</td>
</tr>
<tr>
<td></td>
<td>$VSLOGw. Format (p. 325)</td>
</tr>
<tr>
<td></td>
<td>$VSLOGRw. Format (p. 327)</td>
</tr>
<tr>
<td>CAS</td>
<td>DTWEEKVw. Format (p. 125)</td>
</tr>
<tr>
<td></td>
<td>NLBESTw. Format (p. 152)</td>
</tr>
<tr>
<td></td>
<td>NLDATExw. Format (p. 154)</td>
</tr>
<tr>
<td></td>
<td>NLDATELw. Format (p. 155)</td>
</tr>
<tr>
<td></td>
<td>NLDATENw. Format (p. 156)</td>
</tr>
<tr>
<td></td>
<td>NLDATESw. Format (p. 162)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>NLDATEmw. Format (p. 163)</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the value as the date and the day of the week.</td>
</tr>
<tr>
<td>NLDATENWmw. Format (p. 165)</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the day of the week.</td>
</tr>
<tr>
<td>NLDATEYMWfw. Format (p. 166)</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the name of the month.</td>
</tr>
<tr>
<td>NLDATEYMLw. Format (p. 167)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the month and year.</td>
</tr>
<tr>
<td>NLDATEYMMWw. Format (p. 168)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date values as the month and year with abbreviations.</td>
</tr>
<tr>
<td>NLDATEYMSw. Format (p. 169)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date and year.</td>
</tr>
<tr>
<td>NLDATEYQw. Format (p. 170)</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the quarter.</td>
</tr>
<tr>
<td>NLDATEYQLw. Format (p. 171)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.</td>
</tr>
<tr>
<td>NLDATEYQMw. Format (p. 172)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.</td>
</tr>
<tr>
<td>NLDATEYQSsw. Format (p. 173)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (1–4) with numbers and delimiters.</td>
</tr>
<tr>
<td>NLDATEmw. Format (p. 174)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.</td>
</tr>
<tr>
<td>NLDATEMAPw. Format (p. 175)</td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime with a.m. or p.m.</td>
</tr>
<tr>
<td>NLDATMDTw. Format (p. 176)</td>
<td>Converts the SAS datetime value to the date time value of the specified locale, and then writes the value as the name of the month, day of the month and year.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>NLDATMLw. Format (p. 180)</td>
<td>Converts a SAS datetime value to the datetime string of the specified locale in the long representation of the date.</td>
</tr>
<tr>
<td>NLDATMMw. Format (p. 181)</td>
<td>Converts a SAS datetime value to the datetime string of the specified locale in the medium representation of the date.</td>
</tr>
<tr>
<td>NLDATMMDw. Format (p. 182)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month and the day of the month.</td>
</tr>
<tr>
<td>NLDATMMDLw. Format (p. 182)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the full-length of the month and day of the month.</td>
</tr>
<tr>
<td>NLDATMMDMw. Format (p. 183)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using abbreviations.</td>
</tr>
<tr>
<td>NLDATMMDSw. Format (p. 184)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using numbers and delimiters.</td>
</tr>
<tr>
<td>NLDATMMNw. Format (p. 185)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.</td>
</tr>
<tr>
<td>NLDATMSw. Format (p. 186)</td>
<td>Converts a SAS datetime value to the datetime string of the specified locale in the short representation of the date.</td>
</tr>
<tr>
<td>NLDATMTMw. Format (p. 187)</td>
<td>Converts the time portion of a SAS datetime value to the time-of-day value of the specified locale, and then writes the value as a time of day.</td>
</tr>
<tr>
<td>NLDATMTZw. Format (p. 189)</td>
<td>Converts the time portion of the SAS datetime value to the time of day and time zone of the specified locale.</td>
</tr>
<tr>
<td>NLDATMWw. Format (p. 189)</td>
<td>Converts SAS datetime values to the locale sensitive datetime string as the day of the week and the datetime.</td>
</tr>
<tr>
<td>NLDATMWNw. Format (p. 191)</td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as the day of the week.</td>
</tr>
<tr>
<td>NLDATMWZw. Format (p. 192)</td>
<td>Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.</td>
</tr>
<tr>
<td>NLDATMYMw. Format (p. 193)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the month and year.</td>
</tr>
<tr>
<td>NLDATMYMLw. Format (p. 194)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>NLDATMYMMw. Format (p. 195)</td>
<td></td>
</tr>
<tr>
<td>NLDATMYMSw. Format (p. 196)</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQw. Format (p. 197)</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQMw. Format (p. 199)</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQSsw. Format (p. 200)</td>
<td></td>
</tr>
<tr>
<td>NLDATMYRw. Format (p. 201)</td>
<td></td>
</tr>
<tr>
<td>NLDATMYWw. Format (p. 202)</td>
<td></td>
</tr>
<tr>
<td>NLDATMZw. Format (p. 203)</td>
<td></td>
</tr>
<tr>
<td>NLMNIAEDw.d Format (p. 203)</td>
<td></td>
</tr>
<tr>
<td>NLMNIAUDw.d Format (p. 204)</td>
<td></td>
</tr>
<tr>
<td>NLMNIBGNw.d Format (p. 205)</td>
<td></td>
</tr>
<tr>
<td>NLMNIBRLw.d Format (p. 206)</td>
<td></td>
</tr>
<tr>
<td>NLMNICADw.d Format (p. 207)</td>
<td></td>
</tr>
<tr>
<td>NLMNICHFWw.d Format (p. 208)</td>
<td></td>
</tr>
<tr>
<td>NLMNICNYw.d Format (p. 209)</td>
<td></td>
</tr>
<tr>
<td>NLMNICZKw.d Format (p. 210)</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>NLMNIDKKw.d Format (p. 211)</td>
</tr>
<tr>
<td></td>
<td>NLMNIEEKw.d Format (p. 212)</td>
</tr>
<tr>
<td></td>
<td>NLMNIEGPw.d Format (p. 213)</td>
</tr>
<tr>
<td></td>
<td>NLMNIEURw.d Format (p. 214)</td>
</tr>
<tr>
<td></td>
<td>NLMNIGBPw.d Format (p. 215)</td>
</tr>
<tr>
<td></td>
<td>NLMNIHKDw.d Format (p. 216)</td>
</tr>
<tr>
<td></td>
<td>NLMNIHRKw.d Format (p. 217)</td>
</tr>
<tr>
<td></td>
<td>NLMNIHUFw.d Format (p. 218)</td>
</tr>
<tr>
<td></td>
<td>NLMNIIDRw.d Format (p. 219)</td>
</tr>
<tr>
<td></td>
<td>NLMNIILSw.d Format (p. 220)</td>
</tr>
<tr>
<td></td>
<td>NLMNIINRw.d Format (p. 221)</td>
</tr>
<tr>
<td></td>
<td>NLMNIKRWw.d Format (p. 223)</td>
</tr>
<tr>
<td></td>
<td>NLMNILTLw.d Format (p. 224)</td>
</tr>
<tr>
<td></td>
<td>NLMNILVLw.d Format (p. 225)</td>
</tr>
<tr>
<td></td>
<td>NLMNIMOPw.d Format (p. 226)</td>
</tr>
<tr>
<td></td>
<td>NLMNIMXNW.d Format (p. 227)</td>
</tr>
<tr>
<td></td>
<td>NLMNIMYRW.d Format (p. 228)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>NLMNINOKw.d Format (p. 229)</td>
</tr>
<tr>
<td></td>
<td>NLMNINZDw.d Format (p. 230)</td>
</tr>
<tr>
<td></td>
<td>NLMNIPLNw.d Format (p. 231)</td>
</tr>
<tr>
<td></td>
<td>NLMNIRUBw.d Format (p. 232)</td>
</tr>
<tr>
<td></td>
<td>NLMNISEKw.d Format (p. 233)</td>
</tr>
<tr>
<td></td>
<td>NLMNISGDw.d Format (p. 234)</td>
</tr>
<tr>
<td></td>
<td>NLMNITHBw.d Format (p. 235)</td>
</tr>
<tr>
<td></td>
<td>NLMNITRYw.d Format (p. 236)</td>
</tr>
<tr>
<td></td>
<td>NLMNITWDw.d Format (p. 237)</td>
</tr>
<tr>
<td></td>
<td>NLMNIUSDw.d Format (p. 238)</td>
</tr>
<tr>
<td></td>
<td>NLMNIZARw.d Format (p. 239)</td>
</tr>
<tr>
<td></td>
<td>NLMNLAEDx.d Format (p. 240)</td>
</tr>
<tr>
<td></td>
<td>NLMNLAUDw.d Format (p. 241)</td>
</tr>
<tr>
<td></td>
<td>NLMNLBGNw.d Format (p. 242)</td>
</tr>
<tr>
<td></td>
<td>NLMNLBRLw.d Format (p. 243)</td>
</tr>
<tr>
<td></td>
<td>NLMNLCADw.d Format (p. 244)</td>
</tr>
<tr>
<td></td>
<td>NLMNLCNFw.d Format (p. 245)</td>
</tr>
<tr>
<td></td>
<td>NLMNLCNYw.d Format (p. 246)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>NLMNLCZKw.d Format (p. 247)</td>
</tr>
<tr>
<td></td>
<td>NLMNLDKKw.d Format (p. 248)</td>
</tr>
<tr>
<td></td>
<td>NLMNLEEEKw.d Format (p. 249)</td>
</tr>
<tr>
<td></td>
<td>NLMNLEGPw.d Format (p. 250)</td>
</tr>
<tr>
<td></td>
<td>NLMNLEURw.d Format (p. 251)</td>
</tr>
<tr>
<td></td>
<td>NLMNLGBPw.d Format (p. 252)</td>
</tr>
<tr>
<td></td>
<td>NLMNLHKDw.d Format (p. 253)</td>
</tr>
<tr>
<td></td>
<td>NLMNLHRKw.d Format (p. 254)</td>
</tr>
<tr>
<td></td>
<td>NLMNLUHFw.d Format (p. 255)</td>
</tr>
<tr>
<td></td>
<td>NLMNLIDRw.d Format (p. 256)</td>
</tr>
<tr>
<td></td>
<td>NLMNLILSw.d Format (p. 257)</td>
</tr>
<tr>
<td></td>
<td>NLMNLRw.d Format (p. 258)</td>
</tr>
<tr>
<td></td>
<td>NLMNLJPYw.d Format (p. 259)</td>
</tr>
<tr>
<td></td>
<td>NLMNLKRWw.d Format (p. 260)</td>
</tr>
<tr>
<td></td>
<td>NLMNLLLTLw.d Format (p. 261)</td>
</tr>
<tr>
<td></td>
<td>NLMNLLLVLw.d Format (p. 262)</td>
</tr>
<tr>
<td></td>
<td>NLMNLNMPw.d Format (p. 263)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>NLMNLMXNw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 264)</td>
</tr>
<tr>
<td></td>
<td>NLMNLMYRw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 265)</td>
</tr>
<tr>
<td></td>
<td>NLMNLNOKw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 266)</td>
</tr>
<tr>
<td></td>
<td>NLMNLNZDw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 267)</td>
</tr>
<tr>
<td></td>
<td>NLMNLPLNw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 268)</td>
</tr>
<tr>
<td></td>
<td>NLMNLRUBw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 269)</td>
</tr>
<tr>
<td></td>
<td>NLMNLSEKw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 270)</td>
</tr>
<tr>
<td></td>
<td>NLMNLSGDw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 271)</td>
</tr>
<tr>
<td></td>
<td>NLMNLTHBw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 272)</td>
</tr>
<tr>
<td></td>
<td>NLMNLTRYw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 273)</td>
</tr>
<tr>
<td></td>
<td>NLMNLTWDw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 274)</td>
</tr>
<tr>
<td></td>
<td>NLMNLUSDw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 275)</td>
</tr>
<tr>
<td></td>
<td>NLMNLZARw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 276)</td>
</tr>
<tr>
<td></td>
<td>NLMNYw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 277)</td>
</tr>
<tr>
<td></td>
<td>NLMNYIw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 279)</td>
</tr>
<tr>
<td></td>
<td>NLNUMw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 281)</td>
</tr>
<tr>
<td></td>
<td>NLNUMIw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 282)</td>
</tr>
<tr>
<td></td>
<td>NLPCTw.d Format</td>
</tr>
<tr>
<td></td>
<td>(p. 283)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>NLPCTlw.d Format (p. 285)</td>
</tr>
<tr>
<td></td>
<td>NLPCTNw.d Format (p. 286)</td>
</tr>
<tr>
<td></td>
<td>NLPCTPw.d Format (p. 287)</td>
</tr>
<tr>
<td></td>
<td>NLPVALUEw.d Format (p. 288)</td>
</tr>
<tr>
<td></td>
<td>NLSTRMONw.d Format (p. 290)</td>
</tr>
<tr>
<td></td>
<td>NLSTRQTRw.d Format (p. 291)</td>
</tr>
<tr>
<td></td>
<td>NLSTRWKw.d Format (p. 292)</td>
</tr>
<tr>
<td></td>
<td>NLTIMAPw. Format (p. 294)</td>
</tr>
<tr>
<td></td>
<td>NLTIMEw. Format (p. 295)</td>
</tr>
<tr>
<td></td>
<td>NLTIMELw. Format (p. 296)</td>
</tr>
<tr>
<td></td>
<td>NLTIMEMw. Format (p. 298)</td>
</tr>
<tr>
<td></td>
<td>NLTIMESw. Format (p. 299)</td>
</tr>
<tr>
<td></td>
<td>WEEKUw. Format (p. 328)</td>
</tr>
<tr>
<td></td>
<td>WEEKVw. Format (p. 330)</td>
</tr>
<tr>
<td></td>
<td>WEEKWw. Format (p. 331)</td>
</tr>
<tr>
<td></td>
<td>YYWEEKUw. Format (p. 334)</td>
</tr>
<tr>
<td></td>
<td>YYWEEKVw. Format (p. 336)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>YYWEEKWw. Format (p. 338)</td>
</tr>
<tr>
<td>Character</td>
<td>$EBCDICw. Format (p. 126)</td>
</tr>
<tr>
<td></td>
<td>$UCS2Bw. Format (p. 300)</td>
</tr>
<tr>
<td></td>
<td>$UCS2BEw. Format (p. 301)</td>
</tr>
<tr>
<td></td>
<td>$UCS2Lw. Format (p. 303)</td>
</tr>
<tr>
<td></td>
<td>$UCS2LEw. Format (p. 304)</td>
</tr>
<tr>
<td></td>
<td>$UCS2Xw. Format (p. 305)</td>
</tr>
<tr>
<td></td>
<td>$UCS2XEw. Format (p. 307)</td>
</tr>
<tr>
<td></td>
<td>$UCS4Bw. Format (p. 308)</td>
</tr>
<tr>
<td></td>
<td>$UCS4BEw. Format (p. 309)</td>
</tr>
<tr>
<td></td>
<td>$UCS4Lw. Format (p. 311)</td>
</tr>
<tr>
<td></td>
<td>$UCS4LEw. Format (p. 312)</td>
</tr>
<tr>
<td></td>
<td>$UCS4Xw. Format (p. 313)</td>
</tr>
<tr>
<td></td>
<td>$UCS4XEw. Format (p. 315)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td>$UESCw. Format (p. 316)</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode escape (UESC) representation.</td>
</tr>
<tr>
<td>$UESCEw. Format (p. 317)</td>
<td>Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UNCRw. Format (p. 319)</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in numeric character representation (NCR).</td>
</tr>
<tr>
<td>$UNCREw. Format (p. 320)</td>
<td>Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UPARENw. Format (p. 321)</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation.</td>
</tr>
<tr>
<td>$UPARENEw. Format (p. 322)</td>
<td>Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UTF8Xw. Format (p. 323)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in universal transformation format (UTF-8) encoding.</td>
</tr>
<tr>
<td>$UTF8XEw. Format (p. 324)</td>
<td>Processes a character string that is in universal transformation format (UTF-8), and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Date</td>
<td>NENGOw. Format (p. 151)</td>
</tr>
<tr>
<td></td>
<td>NLDATExw. Format (p. 154)</td>
</tr>
<tr>
<td></td>
<td>NLDATELxw. Format (p. 155)</td>
</tr>
<tr>
<td></td>
<td>NLDATEmxw. Format (p. 156)</td>
</tr>
<tr>
<td></td>
<td>NLDATEMDxw. Format (p. 157)</td>
</tr>
<tr>
<td></td>
<td>NLDATEMDLxw. Format (p. 158)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>NLDATEMDMw. Format</td>
<td>p. 159</td>
</tr>
<tr>
<td>NLDATEMDSw. Format</td>
<td>p. 160</td>
</tr>
<tr>
<td>NLDATEMNw. Format</td>
<td>p. 161</td>
</tr>
<tr>
<td>NLDATESw. Format</td>
<td>p. 162</td>
</tr>
<tr>
<td>NLDATEWw. Format</td>
<td>p. 163</td>
</tr>
<tr>
<td>NLDATENnw. Format</td>
<td>p. 165</td>
</tr>
<tr>
<td>NLDATENMw. Format</td>
<td>p. 166</td>
</tr>
<tr>
<td>NLDATENMLw. Format</td>
<td>p. 167</td>
</tr>
<tr>
<td>NLDATENMMw. Format</td>
<td>p. 168</td>
</tr>
<tr>
<td>NLDATENMSw. Format</td>
<td>p. 169</td>
</tr>
<tr>
<td>NLDATENQw. Format</td>
<td>p. 170</td>
</tr>
<tr>
<td>NLDATENQLw. Format</td>
<td>p. 171</td>
</tr>
<tr>
<td>NLDATENQMw. Format</td>
<td>p. 172</td>
</tr>
<tr>
<td>NLDATENQSw. Format</td>
<td>p. 173</td>
</tr>
<tr>
<td>NLDATENRw. Format</td>
<td>p. 174</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td>NLDATEYWw. Format (p. 175)</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the week.</td>
</tr>
<tr>
<td>NLDATMDTw. Format (p. 179)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month, day of the month and year.</td>
</tr>
<tr>
<td>NLDATMMMdw. Format (p. 182)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month and the day of the month.</td>
</tr>
<tr>
<td>NLDATMMMDLw. Format (p. 182)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the full-length of the month and day of the month.</td>
</tr>
<tr>
<td>NLDATMMMDMw. Format (p. 183)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using abbreviations.</td>
</tr>
<tr>
<td>NLDATMMDSw. Format (p. 184)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using numbers and delimiters.</td>
</tr>
<tr>
<td>NLDATMMNw. Format (p. 185)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.</td>
</tr>
<tr>
<td>NLDATMZWZw. Format (p. 192)</td>
<td>Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.</td>
</tr>
<tr>
<td>NLDATMYMLw. Format (p. 194)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.</td>
</tr>
<tr>
<td>NLDATMYMMw. Format (p. 195)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.</td>
</tr>
<tr>
<td>NLDATMYMSw. Format (p. 196)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the month and year with numbers and a delimiter.</td>
</tr>
<tr>
<td>NLDATMYQLw. Format (p. 198)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter value (1–4) and the year.</td>
</tr>
<tr>
<td>NLDATMYQMw. Format (p. 199)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter (1–4) and then the year.</td>
</tr>
<tr>
<td>NLDATMYQS. Format (p. 200)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the quarter (1-4) using numbers and a delimiter.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Date and Time</td>
<td>B8601DXw. Format</td>
</tr>
<tr>
<td></td>
<td>B8601LXw. Format</td>
</tr>
<tr>
<td></td>
<td>DTWEEKVw. Format</td>
</tr>
<tr>
<td></td>
<td>E8601DXw. Format</td>
</tr>
<tr>
<td></td>
<td>E8601LXw. Format</td>
</tr>
<tr>
<td></td>
<td>HDATEw. Format</td>
</tr>
<tr>
<td></td>
<td>HEBDATEw. Format</td>
</tr>
<tr>
<td></td>
<td>JNENGOw.d Format</td>
</tr>
<tr>
<td></td>
<td>JNENGOTw. Format</td>
</tr>
<tr>
<td></td>
<td>JNENGOTW w. Format</td>
</tr>
<tr>
<td></td>
<td>JNENGOWw.d Format</td>
</tr>
<tr>
<td></td>
<td>MINGUOw. Format</td>
</tr>
<tr>
<td></td>
<td>NLDATMw. Format</td>
</tr>
<tr>
<td></td>
<td>NLDATMAPw. Format</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>NLDATMDTw. Format (p. 179)</td>
</tr>
<tr>
<td></td>
<td>NLDATMLLw. Format (p. 180)</td>
</tr>
<tr>
<td></td>
<td>NLDATMMw. Format (p. 181)</td>
</tr>
<tr>
<td></td>
<td>NLDATMMDw. Format (p. 182)</td>
</tr>
<tr>
<td></td>
<td>NLDATMNNw. Format (p. 185)</td>
</tr>
<tr>
<td></td>
<td>NLDATMSw. Format (p. 186)</td>
</tr>
<tr>
<td></td>
<td>NLDATMTMw. Format (p. 187)</td>
</tr>
<tr>
<td></td>
<td>NLDATMTZw. Format (p. 189)</td>
</tr>
<tr>
<td></td>
<td>NLDATMWw. Format (p. 189)</td>
</tr>
<tr>
<td></td>
<td>NLDATMNw. Format (p. 191)</td>
</tr>
<tr>
<td></td>
<td>NLDATMWZw. Format (p. 192)</td>
</tr>
<tr>
<td></td>
<td>NLDATMYMw. Format (p. 193)</td>
</tr>
<tr>
<td></td>
<td>NLDATMYQw. Format (p. 197)</td>
</tr>
<tr>
<td></td>
<td>NLDATMYRw. Format (p. 201)</td>
</tr>
<tr>
<td></td>
<td>NLDATMYWw. Format (p. 202)</td>
</tr>
<tr>
<td></td>
<td>NLDATMZw. Format (p. 203)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>B8601LXw. Format (p. 117)</td>
</tr>
<tr>
<td></td>
<td>B8601TXw. Format (p. 119)</td>
</tr>
<tr>
<td></td>
<td>E8601DXw. Format (p. 128)</td>
</tr>
<tr>
<td></td>
<td>E8601LXw. Format (p. 129)</td>
</tr>
<tr>
<td></td>
<td>E8601TXw. Format (p. 131)</td>
</tr>
<tr>
<td>Numeric</td>
<td>BESTDOTXw. Format (p. 120)</td>
</tr>
<tr>
<td></td>
<td>EUROw.d Format (p. 132)</td>
</tr>
<tr>
<td></td>
<td>EUROXw.d Format (p. 135)</td>
</tr>
<tr>
<td></td>
<td>NLBESTw. Format (p. 152)</td>
</tr>
<tr>
<td></td>
<td>NLMNIAEDw.d Format (p. 203)</td>
</tr>
<tr>
<td></td>
<td>NLMNIAUDw.d Format (p. 204)</td>
</tr>
<tr>
<td></td>
<td>NLMNIBGNw.d Format (p. 205)</td>
</tr>
<tr>
<td></td>
<td>NLMNIBRLw.d Format (p. 206)</td>
</tr>
<tr>
<td></td>
<td>NLMNICADw.d Format (p. 207)</td>
</tr>
<tr>
<td></td>
<td>NLMNICHFw.d Format (p. 208)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>NLMNICNYw.d Format (p. 209)</td>
<td>Writes the monetary format of the international expression for China.</td>
</tr>
<tr>
<td>NLMNICZKw.d Format (p. 210)</td>
<td>Writes the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td>NLMNIDKKw.d Format (p. 211)</td>
<td>Writes the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td>NLMNIEEKw.d Format (p. 212)</td>
<td>Writes the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td>NLMNIEGPw.d Format (p. 213)</td>
<td>Writes the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td>NLMNIEEUROW.d Format (p. 214)</td>
<td>Writes the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td>NLMNIGBPw.d Format (p. 215)</td>
<td>Writes the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td>NLMNIHKDw.d Format (p. 216)</td>
<td>Writes the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td>NLMNIHRKw.d Format (p. 217)</td>
<td>Writes the monetary format of the international expression for Croatia.</td>
</tr>
<tr>
<td>NLMNIHUFw.d Format (p. 218)</td>
<td>Writes the monetary format of the international expression for Hungary.</td>
</tr>
<tr>
<td>NLMNIIDRW.d Format (p. 219)</td>
<td>Writes the monetary format of the international expression for Indonesia.</td>
</tr>
<tr>
<td>NLMNIIISw.d Format (p. 220)</td>
<td>Writes the monetary format of the international expression for Israel.</td>
</tr>
<tr>
<td>NLMNIIINRw.d Format (p. 221)</td>
<td>Writes the monetary format of the international expression for India.</td>
</tr>
<tr>
<td>NLMNIJPYW.d Format (p. 222)</td>
<td>Writes the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td>NLMNIKRWW.d Format (p. 223)</td>
<td>Writes the monetary format of the international expression for South Korea.</td>
</tr>
<tr>
<td>NLMNILTLW.d Format (p. 224)</td>
<td>Writes the monetary format of the international expression for Lithuania.</td>
</tr>
<tr>
<td>NLMNILVLLW.d Format (p. 225)</td>
<td>Writes the monetary format of the international expression for Latvia.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>NLMNIMOPw.d Format (p. 226)</td>
</tr>
<tr>
<td></td>
<td>NLMNIMXNw.d Format (p. 227)</td>
</tr>
<tr>
<td></td>
<td>NLMNIMYRw.d Format (p. 228)</td>
</tr>
<tr>
<td></td>
<td>NLMNINOKw.d Format (p. 229)</td>
</tr>
<tr>
<td></td>
<td>NLMNINZDw.d Format (p. 230)</td>
</tr>
<tr>
<td></td>
<td>NLMNIPLNw.d Format (p. 231)</td>
</tr>
<tr>
<td></td>
<td>NLMNIRUBw.d Format (p. 232)</td>
</tr>
<tr>
<td></td>
<td>NLMNISEKw.d Format (p. 233)</td>
</tr>
<tr>
<td></td>
<td>NLMNISGDw.d Format (p. 234)</td>
</tr>
<tr>
<td></td>
<td>NLMNITHBw.d Format (p. 235)</td>
</tr>
<tr>
<td></td>
<td>NLMNITRYw.d Format (p. 236)</td>
</tr>
<tr>
<td></td>
<td>NLMNITWDw.d Format (p. 237)</td>
</tr>
<tr>
<td></td>
<td>NLMNIUSDw.d Format (p. 238)</td>
</tr>
<tr>
<td></td>
<td>NLMNIZARw.d Format (p. 239)</td>
</tr>
<tr>
<td></td>
<td>NLMNLAEDx.d Format (p. 240)</td>
</tr>
<tr>
<td></td>
<td>NLMNLAUDw.d Format (p. 241)</td>
</tr>
<tr>
<td></td>
<td>NLMNLBGNw.d Format (p. 242)</td>
</tr>
<tr>
<td></td>
<td>NLMNLBRLw.d Format (p. 243)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>NLMNLCADw.d Format (p. 244)</td>
<td></td>
</tr>
<tr>
<td>NLMNLCADCHFw.d Format (p. 245)</td>
<td></td>
</tr>
<tr>
<td>NLMNLCNYw.d Format (p. 246)</td>
<td></td>
</tr>
<tr>
<td>NLMNLCZKw.d Format (p. 247)</td>
<td></td>
</tr>
<tr>
<td>NLMNLDKKw.d Format (p. 248)</td>
<td></td>
</tr>
<tr>
<td>NLMNLEEKWw.d Format (p. 249)</td>
<td></td>
</tr>
<tr>
<td>NLMNLEGpw.d Format (p. 250)</td>
<td></td>
</tr>
<tr>
<td>NLMNLEURw.d Format (p. 251)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSGBPw.d Format (p. 252)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSHKDw.d Format (p. 253)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSHKRw.d Format (p. 254)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSUFw.d Format (p. 255)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSIDRw.d Format (p. 256)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSILSw.d Format (p. 257)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSINRw.d Format (p. 258)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSJPYw.d Format (p. 259)</td>
<td></td>
</tr>
<tr>
<td>NLMNLSKRw.d Format (p. 260)</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td>NLMNLLTLw.d Format (p. 261)</td>
</tr>
<tr>
<td></td>
<td>NLMNLLVVLw.d Format (p. 262)</td>
</tr>
<tr>
<td></td>
<td>NLMNLMOPw.d Format (p. 263)</td>
</tr>
<tr>
<td></td>
<td>NLMNLMXNw.d Format (p. 264)</td>
</tr>
<tr>
<td></td>
<td>NLMNLMYRw.d Format (p. 265)</td>
</tr>
<tr>
<td></td>
<td>NLMNLNOKw.d Format (p. 266)</td>
</tr>
<tr>
<td></td>
<td>NLMNLNZDw.d Format (p. 267)</td>
</tr>
<tr>
<td></td>
<td>NLMNLPLNw.d Format (p. 268)</td>
</tr>
<tr>
<td></td>
<td>NLMNRUBw.d Format (p. 269)</td>
</tr>
<tr>
<td></td>
<td>NLMNLSEKw.d Format (p. 270)</td>
</tr>
<tr>
<td></td>
<td>NLMNSGDw.d Format (p. 271)</td>
</tr>
<tr>
<td></td>
<td>NLMNLTBDw.d Format (p. 272)</td>
</tr>
<tr>
<td></td>
<td>NLMNLTRYw.d Format (p. 273)</td>
</tr>
<tr>
<td></td>
<td>NLMNLTDw.d Format (p. 274)</td>
</tr>
<tr>
<td></td>
<td>NLMNUSDw.d Format (p. 275)</td>
</tr>
<tr>
<td></td>
<td>NLMNLZARw.d Format (p. 276)</td>
</tr>
<tr>
<td></td>
<td>NLMNYw.d Format (p. 277)</td>
</tr>
<tr>
<td></td>
<td>NLMNYw.d Format (p. 279)</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td>NLNUMw.d Format (p. 281)</td>
<td>Writes the numeric format of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLNUMIw.d Format (p. 282)</td>
<td>Writes the numeric format of the international expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCTw.d Format (p. 283)</td>
<td>Writes percentage data of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCTIw.d Format (p. 285)</td>
<td>Writes percentage data of the international expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCTNw.d Format (p. 286)</td>
<td>Produces percentages, using a minus sign for negative values.</td>
</tr>
<tr>
<td>NLPCTPw.d Format (p. 287)</td>
<td>Writes locale-specific numeric values as percentages.</td>
</tr>
<tr>
<td>NLPVALUEw.d Format (p. 288)</td>
<td>Writes p-values of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLSTRMONw.d Format (p. 290)</td>
<td>Writes the month name in the specified locale.</td>
</tr>
<tr>
<td>NLSTRQTRw.d Format (p. 291)</td>
<td>Writes a numeric value as the quarter-of-the-year in the specified locale.</td>
</tr>
<tr>
<td>NLSTRWKw.d Format (p. 292)</td>
<td>Writes a numeric value as the day-of-the-week in the specified locale.</td>
</tr>
<tr>
<td>YENw.d Format (p. 333)</td>
<td>Writes numeric values with yen signs, commas, and decimal points.</td>
</tr>
<tr>
<td>Time</td>
<td>B8601TXw. Format (p. 119)</td>
</tr>
<tr>
<td></td>
<td>E8601TXw. Format (p. 131)</td>
</tr>
<tr>
<td></td>
<td>NLDATMTMw. Format (p. 187)</td>
</tr>
<tr>
<td></td>
<td>NLDATMTZw. Format (p. 189)</td>
</tr>
<tr>
<td></td>
<td>NLTIMAPw. Format (p. 294)</td>
</tr>
<tr>
<td></td>
<td>NLTIMEw. Format (p. 295)</td>
</tr>
</tbody>
</table>
Dictionary

B8601DXw. Format

Adjusts a Coordinated Universal Time (UTC) datetime value to the user’s local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone basic notation `yyyymmddThhmmss +hhmm`.

**Categories:** Date and Time

**Alignment:** Left

**Restriction:** This format is not supported in a DATA step that runs in CAS.

**Supports:** ISO 8601 Element 5.4.1, complete representation

**Syntax**

`B8601DXw`.

**Syntax Description**

`w`

specifies the width of the output field.

**Default** 26

**Range** 20–35

**Details**

UTC values specify a date and a time that are based on the zero meridian in Greenwich, England. Using this format, SAS converts a datetime value to the UTC value and determines the user local date and time by using the value of the `TIMEZONE=` system option. If the `TIMEZONE=` option is not set, then the user local date and time are based on the local date and time. The B8601DX format writes SAS datetime values by using the following ISO 8601 basic datetime notation:

- `yyyymmddThhmmss+hhmm`

  - `yyyy` is a four-digit year.
  - `mm` is a two-digit month (zero padded) between 01 and 12.
  - `dd` is a two-digit day of the month (zero padded) between 01 and 31.
  - `hh` is a two-digit hour (zero padded) between 00 and 23.
**mm**

is a two-digit minute (zero padded) between 00 and 59.

**ss**

is a two-digit second (zero padded) between 00 and 59.

**+|–hhmm**

is an hour and minute signed offset from zero meridian time. The offset must be **+|–hhmm** (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +0200 indicates a two-hour time difference to the east of the zero meridian, and –0600 indicates a six-hour time difference to the west of the zero meridian.

**Restriction:** The shorter form **+|–hh** is not supported.

### Example

The first example uses the local time to determine the time and the time zone offset. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>t='01Feb2013T12:34:56'dt ;</td>
<td>20130201T073456-0500</td>
</tr>
<tr>
<td>put t b8601dx.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options timezone='America/Adak';</td>
<td></td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>t='01Feb2013T12:34:56'dt ;</td>
<td>20130201T023456-1000</td>
</tr>
<tr>
<td>put t b8601dx.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

“Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations” in SAS Formats and Informats: Reference

---

**B8601LXw. Format**

Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 basic notation yyyyymmddThhmmss+|–hhmm.

**Categories:** Date and Time

**ISO 8601**

**Alignment:** Right

**Restriction:** This format is not supported in a DATA step that runs in CAS.

**Supports:** ISO 8601 Elements 5.3.3 and 5.3.4.2
Syntax

B8601LXw

Syntax Description

w specifies the width of the output field.

Default 26
Range 20–35

Details

The B8601LX format writes datetime values without making any adjustments, and appends the UTC time zone offset for the local SAS session by using the ISO 8601 basic datetime notation:

- yyyyymmddThhmmss+|–hhmm

- yyyy is a four-digit year.
- mm is a two-digit month (zero padded) between 01 and 12.
- dd is a two-digit day of the month (zero padded) between 01 and 31.
- hh is a two-digit hour (zero padded) between 00 and 23.
- mm is a two-digit minute (zero padded) between 00 and 59.
- ss is a two-digit second (zero padded) between 00 and 59.
- +|–hhmm is an hour and minute signed offset from zero meridian time. The offset must be +|–hhmm (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +0200 indicates a two-hour time difference to the east of the zero meridian, and –0600 indicates a six-hour time difference to the west of the zero meridian.

Restriction: The shorter form +|–hh is not supported.

Example

This PUT statement writes the time for the Eastern Standard time zone:

blx='01Feb2013T12:34:56'dt;
put blx b8601lx.;

<table>
<thead>
<tr>
<th>Value of blx</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1675341296</td>
<td>20130201T123456-0500</td>
</tr>
</tbody>
</table>
See Also
"Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations" in SAS Formats and Informats: Reference

B8601TXw. Format
Adjusts a Coordinated Universal Time (UTC) value to the user's local time. Then, writes the local time by using the ISO 8601 basic time notation $hhmmsst+-hhmm$.

Categories: Time
ISO 8601

Alignment: Right

Restriction: This format is not supported in a DATA step that runs in CAS.

Supports: ISO 8601 Elements 5.3.3 and 5.3.4

Syntax
B8601TXw.

Syntax Description

$w$

specifies the width of the output field.

Default 14

Range 9–20

Details
UTC values specify a time based on the zero meridian in Greenwich, England. Using this format, SAS converts a time value to the UTC value and determines the user local time by using the TIMEZONE= system option. If the TIMEZONE= option is not set, then the user local time is based on the local time. The B8601TX format writes SAS datatime values by using the following ISO 8601 basic time notation:

- $hhmmsst+-hhmm$

  - $hh$
    - is a two-digit hour (zero padded) between 00 and 23.
  
  - $mm$
    - is a two-digit minute (zero padded) between 00 and 59.
  
  - $ss$
    - is a two-digit second (zero padded) between 00 and 59.
  
  - $+-hhmm$
    - is an hour and minute signed offset from zero meridian time. The offset must be $+-hhmm$ (that is, + or – and four characters).
Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +0200 indicates a two-hour time difference to the east of the zero meridian, and –0600 indicates a six-hour time difference to the west of the zero meridian.

Restriction: The shorter form +|–hh is not supported.

When SAS reads a UTC time by using the B8601TZ informat, and the adjusted time is greater than 24 hours or less than 00 hours, SAS adjusts the value so that the time is between 000000 and 240000. If the B8601TX format attempts to format a time outside of this time range, the time is formatted with asterisks to indicate that the value is out of range.

Example

The first example uses the local time to determine the time and the time zone offset. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; t='12:34:56't; put t b8601tx.; run;</td>
<td>073456-0500</td>
</tr>
<tr>
<td>options timezone='America/Adak'; data <em>null</em>; t='12:34:56't; put t b8601tx.; run;</td>
<td>023456-1000</td>
</tr>
</tbody>
</table>

See Also

"Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations" in SAS Formats and Informats: Reference

BESTDOTXw. Format

Specifies that SAS choose the best notation and use a dot as a decimal separator.

Category: Numeric
Alignment: Right
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

BESTDOTXw.
Syntax Description

\( w \)

specifies the width of the output field.

Default 12

Range 1–32

Tip If you print numbers between 0 and .01 exclusively, use a field width of at least 7 to avoid excessive rounding. If you print numbers between 0 and -01 exclusively, use a field width of at least 8.

Details

If the NLDECSEPARATOR system option is disabled, the BEST\(w\) and BESTDOTX\(w\) formats process data the same way. If the NLDECSEPARATOR system option is enabled, then the results from the BEST and BESTDOTX formats are different. See the following table to understand the differences:

<table>
<thead>
<tr>
<th>LOCALE option</th>
<th>Default decimal separator character for the locale</th>
<th>NLDECSEPARATOR option</th>
<th>Separator character used by BEST(w)</th>
<th>Separator character used by BESTDOTX(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en_US</td>
<td>Dot</td>
<td>Disabled (default)</td>
<td>Dot</td>
<td>Dot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enabled</td>
<td>Dot</td>
<td>Dot</td>
</tr>
<tr>
<td>fr_FR</td>
<td>Comma</td>
<td>Disabled (default)</td>
<td>Dot</td>
<td>Dot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enabled</td>
<td>Comma</td>
<td>Dot</td>
</tr>
</tbody>
</table>

Example

The following code produces results using the BESTDOTX format with the NLSDECSEPARATOR option enabled and the locale set to fr_FR.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options nld locale=fr_FR;</td>
<td>1,2</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>1.2</td>
</tr>
<tr>
<td>x=1.2;</td>
<td></td>
</tr>
<tr>
<td>call symput('macx', put(x, BESTDOTX.));</td>
<td></td>
</tr>
<tr>
<td>put x; /* Result is printed as &quot;1,2&quot; */</td>
<td></td>
</tr>
<tr>
<td>y=put(x, BESTDOTX.); /*Result is printed as &quot;1.2 */</td>
<td>run;</td>
</tr>
<tr>
<td>&amp;put &amp;macx;</td>
<td>1.2</td>
</tr>
</tbody>
</table>
See Also

- "BESTw. Format" in *SAS Formats and Informats: Reference*
- "NLDECSEPARATOR System Option" on page 705

### $BIDlw. Format

Converts between a logically ordered string and a visually ordered string, by reversing the order of Hebrew and Arabic characters while preserving the order of Latin words and numbers.

**Category:** BIDI Text Handling  
**Alignment:** Left  
**Restriction:** This format is not supported in a DATA step that runs in CAS.

#### Syntax

```
$BIDlw.
```

#### Syntax Description

- `w` specifies the width of the output field.

**Default**  
1 if `w` is not specified  
**Range**  
1–32767

#### Details

In the Windows operating environment, Hebrew and Arabic text is stored in logical order. The text is stored in the order in which it is written and not necessarily as it is displayed. However, in other operating environments, Hebrew text is stored in the same order it is displayed. SAS users can encounter Hebrew and Arabic text that is reversed. Such situations can occur when you use SAS/CONNECT or other software to transfer SAS data sets or reports with Hebrew and Arabic text from a visual operating environment to a logical one. The $BIDI format is a format that reverses Hebrew and Arabic text while maintaining the order of numbers and Latin-1 words.

**Operating Environment Information:** In mainframe operating environments, this format is designed to work with NewCode Hebrew and Arabic. Some mainframe operating environments might experience unsatisfactory results, because they use
the OldCode Hebrew or Arabic encoding. There is a hotfix for this encoding on SAS Institute’s Web site.

Comparisons

The $BIDIw$ format performs a reversing function similar to the $REVERJw$ format, which writes character data in reverse order and preserves blanks. $BIDIw$ behaves in the following way:

- $BIDIw$ reverses the order of words and numbers in a specified string, preserving blanks. Latin-1 words and numbers themselves are not reversed, only their order in the string.
- When $BIDI$ encounters a word consisting of Hebrew or Arabic characters in the text string, the characters in the Hebrew or Arabic word are reversed. The position of the Hebrew or Arabic word is reversed in the string.

Example

This example demonstrates how $BIDIw$ reverses Hebrew characters. The Hebrew is reversed in the string. The Hebrew characters in the words are also reversed.

```sas
data;
  a = "שלאות נא צ";
  b1 = put (a,$bidi20.);
  put b1;
  b2 = put (b,$bidi20.);
  put b2;
run;
```

The following lines are written to the SAS log:

```
 b1=123 נא צ השלאות
 b2= נא צ 123
```

$CPTDWw. Format

Processes a character string that is in Hebrew text, encoded in IBM-PC (cp862), and then writes the character string in Windows Hebrew encoding (cp 1255).

Category: Hebrew Text Handling
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$CPTDWw$

Syntax Description

- $w$ specifies the width of the output field.
The \$CPTDWw. format performs processing that is the opposite of the \$CPTWDw. format.

Example
The following example uses the input value of “808182x.”

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $cptdw3.;</td>
<td>1234</td>
</tr>
</tbody>
</table>

See Also

Format:
- “\$CPTWDw. Format” on page 124

Informats:
- “\$CPTDWw. Informat” on page 535
- “\$CPTWDw. Informat” on page 536

\$CPTWDw. Format
Processes a character string that is encoded in Windows (cp1255), and then writes the character string in Hebrew DOS (cp862) encoding.

Category: Hebrew Text Handling
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax
\$CPTWDw.

Syntax Description
- \( w \) specifies the width of the output field.
Default: 200

Range: 1–32767

Comparisons
The $CPTWDw. format performs processing that is the opposite of the $CPTDWw. format.

Example
The following example uses the input value of "1234567890".

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $cptwd1.;</td>
<td>( \oplus ),</td>
</tr>
</tbody>
</table>

See Also

**Format:**
- "$CPTDWw. Format" on page 123

**Informats:**
- "$CPTDWw. Informat" on page 535
- "$CPTWDw. Informat" on page 536

### DTWEEKVw. Format

Writes a week and datetime number in decimal format by using the V algorithm.

**Categories:** CAS

Date and Time

**Alignment:** Left

**Syntax**

```plaintext
DTWEEKVw:
```

**Syntax Description**

- \( w \) specifies the width of the output field.
Details

This table has information on widths, formats, and examples:

<table>
<thead>
<tr>
<th>Width</th>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>3–4</td>
<td>WWW</td>
<td>w01</td>
</tr>
<tr>
<td>5–6</td>
<td>yyWww</td>
<td>03w01</td>
</tr>
<tr>
<td>7–8</td>
<td>yyWwwdd</td>
<td>03w0101</td>
</tr>
<tr>
<td>9–11</td>
<td>yyyyWwwdd</td>
<td>2003w0101</td>
</tr>
<tr>
<td>12–14</td>
<td>yyyyWwwdd:hh</td>
<td>2003w0101:09</td>
</tr>
<tr>
<td>15–17</td>
<td>yyyyWwwdd:hh:mm</td>
<td>2003w0101:09:10</td>
</tr>
<tr>
<td>18–19</td>
<td>yyyyWwwdd:hh:mm:ss</td>
<td>2003w0101:09:10:56</td>
</tr>
<tr>
<td>20–200</td>
<td>yyyy-Www-dd:hh:mm:ss</td>
<td>2003-w01-01:09:10:56</td>
</tr>
</tbody>
</table>

Example

Here is an example of the DTWEEKVw. format:

```plaintext
DATA _NULL_;  
t='01Feb2017:1:2:3'dt;  
put t= DTWEEKV30.;  
RUN;
```

$EBCDICw. Format

Converts native format character data to EBCDIC representation.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.
Note: UTF-8 is the only supported session encoding. $EBCDIC format works only for 7-bit ASCII characters.
Syntax

\$EBCDICw.

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 1

Range 1–32767

Details

If EBCDIC is the native format, no conversion occurs.

On ASCII systems, the \$EBCDICw. format is based on the default encoding value of the \texttt{LOCALE=} option that is specified when SAS starts. For example, if the locale was set to \texttt{en-US} locale, the default encoding that is used by the \$EBCDICw. format is Open\_ed-1047. If the locale is \texttt{de-DE} (German_Germany), the default encoding that is used by the \$EBCDICw. format is Open\_ed–1141. For a list of locales and encoding values, see Table 21.2 on page 819.

You can specify the translation table that is used to map characters between EBCDIC and ASCII by using the \texttt{MAPEBCDIC2ASCII=} system option. For more information, see “MAPEBCDIC2ASCII= System Option” on page 704.

Comparisons

- On ASCII systems, \$EBCDICw. converts ASCII character data to EBCDIC.
- On all other systems, \$EBCDICw. behaves like the \$CHARw. format.

Example

```sas
data one;
  input x $;
  datalines;
  ABC
  {ä}
  [ø]
  ;
run;

data two;
  set one;
  y=put(x,$ebcdic3.);
  put y $HEX6.;
run;
```
Table 10.1  Output Locale Values for American English, French, and German

<table>
<thead>
<tr>
<th>Value of Name</th>
<th>Locale Value Is en_US</th>
<th>Locale Value Is fr_FR</th>
<th>Locale Value Is de_DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>C1C2C3</td>
<td>C1C2C3</td>
<td>C1C2C3</td>
</tr>
<tr>
<td>{ä}</td>
<td>C043D0</td>
<td>514354</td>
<td>43C0DC</td>
</tr>
<tr>
<td>[@]</td>
<td>AD7CBD</td>
<td>9044B5</td>
<td>63B5FC</td>
</tr>
</tbody>
</table>

The results are shown as hexadecimal representations of EBCDIC codes for characters. Each pair of hexadecimal characters correspond to one byte of binary data, and each byte corresponds to one character.

E8601DXw. Format

Adjusts a Coordinated Universal Time (UTC) datetime value to the user’s local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone extended notation `yyyy-mm-ddThh:mm:ss+hh:mm`.

Categories: Date and Time
ISO 8601

Alignment: Left

Supports: ISO 8601 Element 5.4.1, complete representation

Syntax

```
E8601DXw.
```

Syntax Description

`w`

specifies the width of the output field.

- Default: 26
- Range: 20–35

Details

UTC values specify a date and time that are based on the zero meridian in Greenwich, England. Using this format, SAS converts a datetime value to the UTC value and determines the user local date and time by using the value of the `TIMEZONE=` system option. If the `TIMEZONE=` option is not set, the user local date and time are based on the local date and time. The E8601DX format writes SAS datetime values by using this ISO 8601 basic datetime notation:

- `yyyy-mm-ddThh:mm:ss+hh:mm`
**yyyy**

is a four-digit year.

**mm**

is a two-digit month (zero padded) between 01 and 12.

**dd**

is a two-digit day of the month (zero padded) between 01 and 31.

**hh**

is a two-digit hour (zero padded) between 00 and 23.

**mm**

is a two-digit minute (zero padded) between 00 and 59.

**ss**

is a two-digit second (zero padded) between 00 and 59.

**+|–hh:mm**

is an hour and minute signed offset from zero meridian time. The offset must be +|–hh:mm (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +02:00 indicates a two-hour time difference to the east of the zero meridian, and –06:00 indicates a six-hour time difference to the west of the zero meridian.

**Restriction:** The shorter form +|–hh is not supported.

### Example

The first example uses the local time to determine the time. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; t='15Sep2018T12:34:56'dt ; put t e8601dx.; run;</td>
<td>2018-09-15T08:34:56-04:00</td>
</tr>
<tr>
<td>options timezone='America/Adak'; data <em>null</em>; t='15Sep2018T12:34:56'dt ; put t e8601dx.; run;</td>
<td>2018-09-15T03:34:56-09:00</td>
</tr>
</tbody>
</table>

### See Also

“Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations” in *SAS Formats and Informats: Reference*
Syntax

\texttt{E8601LXw}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default  \hspace{1cm} 26

Range \hspace{1cm} 20–35

Details

The E8601LX format writes datetime values without making any adjustments, and appends the UTC time zone offset for the local SAS session by using this ISO 8601 basic datetime notation:

\[ \text{yyyy-mm-ddThh:mm:ss+hh:mm} \]

\textit{yyyy}

is a four-digit year.

\textit{mm}

is a two-digit month (zero padded) between 01 and 12.

\textit{dd}

is a two-digit day of the month (zero padded) between 01 and 31.

\textit{hh}

is a two-digit hour (zero padded) between 00 and 23.

\textit{mm}

is a two-digit minute (zero padded) between 00 and 59.

\textit{ss}

is a two-digit second (zero padded) between 00 and 59.

\textit{+|–hh:mm}

is an hour and minute signed offset from zero meridian time. The offset must be \textit{+|–hh:mm} (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +02:00 indicates a two-hour time difference to the east of the zero meridian, and –06:00 indicates a six-hour time difference to the west of the zero meridian.

\textbf{Restriction:} The shorter form \textit{+|–hh} is not supported.

Example

This example writes the time for the Eastern time zone:
data one;
   mydt='15sep2018 T12:34:56'dt;
   put mydt e8601lx.
run;

<table>
<thead>
<tr>
<th>Value of mydt</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1852634096</td>
<td>2018-09-15T12:34:56-05:00</td>
</tr>
</tbody>
</table>

See Also

"Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations" in SAS Formats and Informats: Reference

E8601TXw. Format

Adjusts a Coordinated Universal Time (UTC) value to the user's local time. Then, writes the local time by using the ISO 8601 extended time notation \( hh:mm:ss+|-hh:mm \).

Categories: Time
ISO 8601
Alignment: Right
Supports: ISO 8601 Elements 5.3.3 and 5.3.4

Syntax

\( \text{E8601TX}w \)

Syntax Description

\( w \)

specifies the width of the output field.

Default 14
Range 9–20

Details

UTC values specify a time based on the zero meridian in Greenwich, England. Using this format, SAS converts a time value to the UTC value and determines the user local time by using the value of the TIMEZONE= system option. If the TIMEZONE= option is not set, the user local time is based on the local time. The E8601TX format writes SAS datetime values by using this ISO 8601 basic time notation:

\( hh:mm:ss+|-hh:mm \)

\( hh \)

is a two-digit hour (zero padded) between 00 and 23.
is a two-digit minute (zero padded) between 00 and 59.

SS

is a two-digit second (zero padded) between 00 and 59.

+–hh:mm

is an hour and minute signed offset from zero meridian time. The offset must be +–hhmm (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +0200 indicates a two-hour time difference to the east of the zero meridian, and –0600 indicates a six-hour time difference to the west of the zero meridian.

**Restriction:** The shorter form +–hh is not supported.

When SAS reads a UTC time by using the E8601TZ informat, and the adjusted time is greater than 24 hours or less than 00 hours, SAS adjusts the value so that the time is between 000000 and 240000. If the E8601TX format attempts to format a time outside of this time range, the time is formatted with asterisks to indicate that the value is out of range.

### Example

The first example uses the local time to determine the time and the time zone offset. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; t='12:34:56'; put t e8601tx.; run;</td>
<td>07:34:56-05:00</td>
</tr>
<tr>
<td>options timezone='America/Adak'; data <em>null</em>; t='12:34:56'; put t e8601tx.; run;</td>
<td>02:34:56-10:00</td>
</tr>
</tbody>
</table>

### See Also

*“Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations” in SAS Formats and Informats: Reference*
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

**EURO**\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default: 6

Range: 1-32

Tip: If you want the euro symbol to be part of the output, be sure to choose an adequate width.

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 0

Range: 0-31

Requirement: must be less than \(w\)

Comparisons

- The \(\text{EURO}\text{w}.d\) format is similar to the \(\text{EUROXw}.d\) format, but \(\text{EUROXw}.d\) format reverses the roles of the decimal point and the comma. This convention is common in European countries.
- The \(\text{EURO}\text{w}.d\) format is similar to the \(\text{DOLLARw}.d\) format, except that \(\text{DOLLARw}.d\) format writes a leading dollar sign instead of the euro symbol.

Example

These examples use 1254.71 as the value of amount.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put amount euro10.2;</td>
<td>E1,254.71</td>
</tr>
<tr>
<td>put amount euro5.;</td>
<td>1,255</td>
</tr>
<tr>
<td>put amount euro9.2;</td>
<td>E1,254.71</td>
</tr>
<tr>
<td>put amount euro15.3;</td>
<td>E1,254.710</td>
</tr>
</tbody>
</table>
data _null_;  
  input x;  
  put x euro10.2;  
  put x euro5.;  
  put x euro9.2;  
  put x euro15.3;  
  datalines;  
1254.71  
;  
run;  
SAS Log:  
E1,254.71  
1,255  
E1,254.71  
   E1,254.710  
   /* This code determines the default length. */  
data _null_;  
  input x;  
  put x euro.;  
  datalines;  
1  
22  
333  
4444  
55555  
666666  
7777777  
88888888  
999999999  
1234561234  
;run;  
SAS Log:  
   datalines;  
   E1  
   E22  
   E333  
   E4,444  
   55,555  
   666666  
   7.78E6  
   8.89E7  
   1E9  
   1.23E9  
   NOTE: At least one W.D format was too small for the number to be printed.  
The decimal may be shifted by the "BEST" format.  
   /* This code determines the range. */  
data _null_;  
  input x;  
  put x euro5.;  
  put x euro6.;  
  put x euro7.;  
  put x euro8.;  
  put x euro9.;  
  put x euro9.2;
EUROXw.d Format

Writes numeric values with a leading euro symbol (E), a period that separates every three digits, and a comma that separates the decimal fraction.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Right</td>
</tr>
<tr>
<td>Restriction:</td>
<td>This format is not supported in a DATA step that runs in CAS.</td>
</tr>
</tbody>
</table>

Syntax

```
EUROXw.d
```

Syntax Description

- **w**
  - specifies the width of the output field.

  Default: 6
Tip
If you want the euro symbol to be part of the output, be sure to choose an adequate width.

$d$

specifies the number of digits to the right of the decimal point in the numeric value.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0-31</td>
<td>must be less than $w$</td>
</tr>
</tbody>
</table>

Comparisons

- The EUROX$w.d$ format is similar to the EURO$w.d$ format, but EURO$w.d$ format reverses the roles of the comma and the decimal point. This convention is common in English–speaking countries.
- The EUROX$w.d$ format is similar to the DOLLARX$w.d$ format, except that DOLLARX$w.d$ format writes a leading dollar sign instead of the euro symbol.

Example

These examples use 1254.71 as the value of amount.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put amount eurox10.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox5.;</td>
<td>1.255</td>
</tr>
<tr>
<td>put amount eurox9.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox15.3;</td>
<td>E1.254,710</td>
</tr>
</tbody>
</table>

```sas
data _null_;  
  input x;  
  put x eurox10.2;  
  put x eurox5.;  
  put x eurox9.2;  
  put x eurox15.3;  
  datalines;  
1254.71  
; run;  
SAS Log:  
E1.254,71  
1.255  
E1.254,71  
E1.254,710```
This code determines the default length.

```sas
/* This code determines the default length. */
data _null_
   input x;
   put x eurox.;
datalines;
1
22
333
4444
55555
666666
7777777
88888888
999999999
1234561234
;run;
SAS Log:
   E1
   E22
   E333
   E4.444
   55.555
   666666
   7.78E6
   8.89E7
   1E9
   1.23E9
```

**Note:** At least one W.D format was too small for the number to be printed. The decimal might be shifted by the "BEST" format.

---

See Also

- **Format:**
  - “EUROw.d Format” on page 132

- **Informats:**
  - “EUROw.d Informat” on page 537
  - “EUROXw.d Informat” on page 539

---

### HDATExw. Format

Writes date values in the form `yyyy mmmmm dd` where `dd` is the day-of-the-month, `mmmmmm` represents the month’s name in Hebrew, and `yyyy` is the year.

**Category:** Date and Time

**Alignment:** Right

**Restriction:** This format is not supported in a DATA step that runs in CAS.
Syntax

\texttt{HDATEw.}

Syntax Description

\textit{w} specifies the width of the output field.

\textbf{Note:} Use widths 9, 11, 15, or 17 for the best view.

\begin{itemize}
  \item \textbf{Default:} 17
  \item \textbf{Range:} 9–17
\end{itemize}

Details

The \texttt{HDATEw.} format writes the SAS date value in the form \textit{yy \textit{mmmm} \textit{dd}}:

\begin{itemize}
  \item \textit{yyyy} is the year
  \item \textit{mmmm} is English month name written in Hebrew letters
  \item \textit{dd} is the day-of-the-month
\end{itemize}

Example

The following example uses the input value of 15780, which is the SAS date of March 16, 2003.

\begin{tabular}{|l|l|}
  \hline
  \textbf{Statements} & \textbf{Results} \\
  \hline
  put day hdate9.; & 03 \text{n} \text{n} \text{n} \text{n} 16 \\
  put day hdate11.; & 2003 \text{n} \text{n} \text{n} \text{n} 16 \\
  put day hdate17.; & 2003 \text{n} \text{n} \text{n} \text{n} 16 \\
  \hline
\end{tabular}

See Also

\textbf{Format:}

- “HEBDATEw. Format” on page 139
HEBDATEx. Format

Writs date values according to the Jewish calendar.

Category: Date and Time
Alignment: Right
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

HEBDATEx.

Syntax Description

w

specifies the width of the output field.

Default 16
Range 7–24
Tip When using a non-Hebrew encoding, asterisks appear instead of the Hebrew letters.

Details

The Jewish calendar is a combined solar and lunar calendar. Years are counted from the creation of the world, which according to Jewish history, occurred 3760 years and three months before the commencement of the Christian. You must add 3761, beginning in the autumn of a specified year in the Gregorian calendar to calculate the Hebrew year.

The HEBDATEx. format writes the SAS date value according to the Jewish calendar. The date is written in one of the following formats:

long

default

short

Example

The following example uses the input value of 15780, which is the SAS date of March 16, 2003.
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day hebdatem3. ;</td>
</tr>
<tr>
<td>put day hebdatem6. ;</td>
</tr>
<tr>
<td>put day hebdatem24. ;</td>
</tr>
</tbody>
</table>

### Results

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>י'ג/12/2015</td>
</tr>
<tr>
<td>י'ב שאר תשמ&quot;א</td>
</tr>
<tr>
<td>רא'ת י'ב אדר תשמ&quot;א</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- "HDATEw. Format" on page 137

---

### JNENGOw.d Format

Writes SAS date values as Japanese dates as reign, year, month, and day.

**Category:** Date and Time

**Alignment:** Left

**Restrictions:**
- This format is not supported in a DATA step that runs in CAS.
- This format is supported only in a Japanese environment.

#### Syntax

**JNENGOw.d**

**Optional Arguments**

**w**

- specifies the width of the input field.
- **Default:** 21
- **Range:** 6–200

**d**

- specifies these values:
  - **.0** writes all information.
  - **.1** writes Reign, Year, and Month information.
  - **.2** writes Reign and Year information.
  - **.3** writes Reign information only.
JNENGOw. Format

Writes SAS datetime values as Japanese datetimes as reign, year, month, day, hour, and minute, and with narrow characters for numbers.

Category: Date and Time
Alignment: Left
Restrictions: This format is not supported in a DATA step that runs in CAS.
This format is supported only in a Japanese environment.

Syntax

JNENGOw.

Optional Argument

\( w \)

specifies the width of the input field.

Default 32
Range 27–200
Comparisons

JNENGTOW<w> is similar to JNENGTOW<w> except that JNENGTOW<w> uses narrow characters for numbers.

Example

This example uses the Japanese Japan session encoding. It uses the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003.

data _null_;  
  x = 1361709583 ;  
  put x JNENGT32.;  
  put x JNENGT40.;  
run;

平成15年2月24日12:39
平成15年2月24日12時39分

JNENGTOW<w> Format

Writes SAS datetime values as Japanese datetimes as reign, year, month, day, hour, and minute, and with wide characters for numbers.

Category: Date and Time
Alignment: Left
Restrictions: This format is supported only in a Japanese environment. This format is not supported in a DATA step that runs in CAS.

Syntax

JNENGTOW<w>.

Optional Argument

<w> specifies the width of the input field.

Default 52
Range 49–200

Comparisons

JNENGTOW<w> is similar to JNENGTOW<w> except that JNENGTOW<w> uses wide characters for numbers.
Example

This example uses the Japanese_Japan session encoding. It uses the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003.

```sas
Data _null_; 
  x = 1361709583 ; 
  put x JNENGOTW49.; 
  put x JNENGOTW80.; 
run;
```

平成15年 2月24日 12時39分

---

**JNENGOWw.d Format**

Writes SAS date values as Japanese dates as reign, year, month, and day, and with wide character for numbers.

**Category:** Date and Time

**Alignment:** Left

**Restrictions:**
- This format is not supported in a DATA step that runs in CAS.
- This format runs only in a Japanese environment.

**Syntax**

**JNENGOWw.d**

**Optional Arguments**

- **w**
  - Specifies the width of the input field.
  - Default: 33
  - Range: 6–200

- **d**
  - Specifies these values:
  - .0 writes all information.
  - .1 writes Reign, Year, and Month information.
  - .2 writes Reign and Year information.
  - .3 writes Reign information only.
  - Default: 0
  - Range: 0–3
Comparisons

JNENGOWw.d is similar to JNENGOw.d except that JNENGOWw.d uses wide characters for numbers.

Example

This example uses the Japanese_Japan session encoding. It uses the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

```sas
data _null;
  x = 15760;
  put x JNENGOW.3;
  put x JNENGOW.2;
  put x JNENGOW.1;
  put x JNENGOW.;
run;
```

平成
平成15年
平成15年 2月
平成15年 2月24日

$\text{KANJI}w.$ Format

Adds shift-code data to DBCS data.

Category: DBCS
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$\text{KANJI}w.$

Syntax Description

$w$

specifies the width of the output field.

Range The minimum width of the format is $2 + (\text{length of shift code used on the current DBCSTYPE= setting})^2$

Restriction The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.
Details
The $KANJI format adds shift-code data to DBCS data that does not have shift-code data. If the input data is blank, shift-code data is not added.

The $KANJI format processes host-mainframe data, but $KANJI can be used on other platforms. If you use the $KANJI format on non-EBCDIC (non-modal encoding) hosts, the data does not change.

See Also

Formats:
- “$KANJIXw. Format” on page 145

Informat:
- “$KANJIw. Informat” on page 541
- “$KANJIXw. Informat” on page 542

System Option:
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 689

---

$KANJIXw. Format
Removes shift-code data from DBCS data.

Category: DBCS
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$KANJIXw.

Syntax Description

\(w\)

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Range</th>
<th>The minimum width of the format is 2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.</td>
</tr>
</tbody>
</table>
Details

The $KANJIX format removes shift-code data from DBCS data. The input data length must be $2 + (\text{SO/SI length})\times2$. The data must start with SO and end with SI, unless single-byte data is returned.

The $KANJIX$ format processes host mainframe data, but $KANJIX$ can be used on other platforms. If you use the $KANJIX$ format on non-EBCDIC (non-modal encoding) hosts, the data does not change.

See Also

Format:
- “$KANJIlw. Format” on page 144

Informats:
- “$KANJIlw. Informat” on page 541
- “$KANJIXlw. Informat” on page 542

System Option:
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 689

$LOGVS lw. Format

Processes a character string that is in left-to-right-logical order, and then writes the character string in visual order.

Category: BIDI Text Handling
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$LOGVS lw.

Syntax Description

lw
  specifies the width of the output field.

Default
  200

Range
  1–32767
Details
The $LOGVS\_w. format is used when you store logical-ordered text on a visual server.

Note: If the $LOGVS\_w. format is not accessible, then the Hebrew or Arabic portion of the data is reversed.

Comparisons
The $LOGVS\_w. format performs processing that is the opposite of the $LOGVSR\_w. format.

Example
The following example uses the Hebrew input value of "flight".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12.;</td>
<td>נטעה flight</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "computer".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12.;</td>
<td>ذات computer</td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$LOGVSRw. Format" on page 148

Informats:
- "$LOGVSRw. Informat" on page 544
- "$LOGVSw. Informat" on page 543
$LOGVSRw. Format

Processes a character string that is in right-to-left-logical order, and then writes the character string in visual order.

Category: BIDI Text Handling
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$LOGVSRw.

Syntax Description

w

specifies the width of the output field.

Default 200
Range 1–32767

Details

The $LOGVSRw. format is used when you store logical-ordered text on a visual server. The Hebrew or Arabic portion of the text is reversed if the $LOGVSw. format is not on the server.

Comparisons

The $LOGVSRw. format performs processing that is opposite of the $LOGVSw. format.

Example

The following example uses the Hebrew input value of "_flight".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvsr12.;</td>
<td>flight מטח</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "computer."
Statements | Results
---+----+---

put text $logvsrl2.;

See Also

Formats:
- “$LOGVSw. Format” on page 146

Informats:
- “$LOGVSw. Informat” on page 543
- “$LOGVSRw. Informat” on page 544

MINGUOw. Format
Wries date values as Taiwanese dates in the form yyyyymmdd.

Category: Date and Time
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

MINGUOw.

Syntax Description

w
specifies the width of the output field.

Default: 8

Range: 1–10

Details

The MINGUOw. format writes SAS date values in the form yyyyymmdd, where

yyyy
is an integer that represents the year.
**mm**

is an integer that represents the month.

**dd**

is an integer that represents the day of the month.

The Taiwanese calendar uses 1912 as the base year (01/01/01 is January 1, 1912). Dates before 1912 appear as a series of asterisks. Year values do not roll around after 100 years. Instead, they continue to increase.

### Example

The example table uses the following input values:

- 12054 is the SAS date value that corresponds to January 1, 1993.
- 18993 is the SAS date value that corresponds to January 1, 2012.
- -20088 is the SAS date value that corresponds to January 1, 1905.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x=put(12054,minguo7.);</code></td>
<td><code>x=820101</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(12054,minguo9.);</code></td>
<td><code>x=82/01/01</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(12054,minguo10.);</code></td>
<td><code>x=0082/01/01</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(18993,minguo7.);</code></td>
<td><code>x=1000101</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(18993,minguo9.);</code></td>
<td><code>x=100/01/01</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(18993,minguo10.);</code></td>
<td><code>x=0100/01/01</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(-20088,minguo7.);</code></td>
<td>**********</td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(-20088,minguo9.);</code></td>
<td>**********</td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(-20088,minguo10.);</code></td>
<td>**********</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**

- "MINGUOw. Informat" on page 546
NENGOw. Format

Writes date values as Japanese dates in the form e.ymmd.

Category: Date
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

NENGOw.

Syntax Description

w
  specifies the width of the output field.

  Default  10
  Range    2–10

Details

The NENGOw. format writes SAS date values in the form e.ymmd, where

e  is the first letter of the name of the imperial era (Meiji, Taisho, Showa, Heisei, or Reiwa).

yy  is an integer that represents the year.

mm  is an integer that represents the month.

dd  is an integer that represents the day of the month.

If the width is too small, SAS omits the period.

Example

The example table uses the input value of 15342, which is the SAS date value that corresponds to January 2, 2002.

```
data _null_;
  date=15342;
  put date nengo3.;
  put date nengo6.;
  put date nengo8.;
  put date nengo9.;
  put date nengo10.;
run;
```
NLBEST\(w\). Format

Writes the best numerical notation based on the locale.

**Categories:**
- CAS
- Numeric

**Alignment:** Right

**Syntax**

\texttt{NLBEST\(w\)}

**Syntax Description**

\(w\)

- Specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
<th>Tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1–32</td>
<td></td>
</tr>
</tbody>
</table>

- If you print numbers between 0 and .01 exclusively, then use a field width of at least 7 to avoid excessive rounding. If you print numbers between 0 and -.01 exclusively, use a field width of at least 8.

**Details**

The NLBEST format writes the best numerical value based on the locale's decimal point and the sign mark's location. NLBEST is similar to the BEST format. For more information, see the BEST format in the SAS Formats and Informats: Reference.

**Example**

The following code produces results based on the locale:

```plaintext
options locale=English_UnitedStates;
data test;
x=-1257000;
```
options locale=German_Germany;
data test;
x=-1257000;
put x nlbest6.;
put x nlbest3.;
put "=====";
x=-0.1;
put x nlbest6.;
put x nlbest3.;
put "=====";
x=0.1;
put x nlbest6.;
put x nlbest3.;
put "=====";
x=1257000;
put x nlbest6.;
put x nlbest3.;
run;

-126E4
***
=====
-0.1
-.1
=====
0.1
0.1
=====
1.26E6
1E6
options locale=ar_BH;
data test;
    x=-1257000;
    put x nlbest6.;
    put x nlbest3.;
    put "=====";
    x=-0.1;
    put x nlbest6.;
    put x nlbest3.;
    put "=====";
    x=0.1;
    put x nlbest6.;
    put x nlbest3.;
    put "=====";
    x=1257000;
    put x nlbest6.;
    put x nlbest3.;
run;

-126E4
***
=====
-0.1
-.1
=====
0.1
0.1
=====
1.26E6
1E6

NLDATEmw. Format

Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date.

Categories:
   CAS
   Date

Alignment:
   Left

Syntax

NLDATEmw.

Syntax Description

w
   specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.
Comparisons

NLDATEw. is similar to DATEw. and WORDDATEw., except that NLDATEw. is locale-specific.

Example

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003. The first example specifies the english_UnitedStates locale. The second example specifies the german_Germany locale.

```
options locale=English_UnitedStates;
data test;
  day=15760;
  put day nldate.;
run;

February 24, 2003
```

```
options locale=German_Germany;
data test;
  day=15760;
  put day nldate.;
run;

24. Februar 2003
```

See Also

Formats:
- “NLDATEMNw. Format” on page 161
- “NLDATEWW. Format” on page 163
- “NLDATENUM. Format” on page 165
Syntax

\textbf{NLDATEL}w.

Syntax Description

\textbf{w}

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 18

Range 2–200

Details

NLDATEL writes the date in a long-uniform pattern with the full length of the month and week names.

Example

This example uses the date November 19, 2012.

\begin{verbatim}
data _null_;  dt = dt='19Nov2012:00:00:00'dt;  dy='19Nov2012'd;  put "---- NLDATEL min=2 default=18
max=200 ----";  put dy nldatel.;  put dy nldatel10.;  put dy nldatel12.;  put dy nldatel18.;  put dy nldatel200.;  run;
\end{verbatim}

\begin{verbatim}
---- NLDATEL min=2 default=18 max=200 ----
November 19, 2012
11/19/2012
Nov 19, 2012
November 19, 2012
November 19, 2012
\end{verbatim}

\textbf{NLDATEM}w. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date in a medium-uniform pattern.

Categories: \hspace{1cm} CAS
\hspace{1cm} Date

Alignment: \hspace{1cm} Left
Syntax

**NLDATEM**\(w\)

Syntax Description

\(w\) specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default: 14

Range: 2–200

Details

NLDATEM writes the date in a medium-uniform pattern such as Nov 19, 2012.

Example

This example specifies the date Nov 19, 2012.

```sas
data _null_;  
  dt = dt='19Nov2012:00:00:00'dt;  
  dy='19Nov2012'd;  
  put '+--- NLDATEM min=2 default=14 max=200 ---+';  
  put dy nldatem.;  
  put dy nldatem8.;  
  put dy nldatem14.;  
  put dy nldatem200.;  
run;
```

+--- NLDATEM min=2 default=14 max=200 ---+
Nov 19, 2012
11/19/12
Nov 19, 2012
Nov 19, 2012

--- NLDATEMDw. Format ---

Converts the SAS date value to the date value of the specified locale, and then writes the value as the name of the month and the day of the month.

Categories: CAS

Date

Alignment: Left

Syntax

**NLDATEMD**\(w\).
Syntax Description

\( w \)
specifies the width of the output field.

- **Default**: 16
- **Range**: 6-200

Example

This example specifies the `en_US` locale option.

```sas
option locale=en_US;
data _null_; 
  dy = date();
  put dy nldatemd.;
run;
```

April 03

See Also

- **Format**: “NLDATEYMw. Format” on page 166

---

**NLDATEMDLw. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.

**Categories**: CAS, Date

**Alignment**: Left

Syntax

\( \text{NLDATEMDL} w. \)

Syntax Description

\( w \)
specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default**: 12
- **Range**: 5–200
Details

NLDATEMDL writes the date in a long-uniform pattern with the full length of the month and the day such as November 19.

Example

The following example specifies the date of November 19, 2012.

```
data _null_
  dt='19Nov2012:00:00:00'dt;
  dy='19Nov2012'd;
  put '+'--- NLDATEMDL min=5 default=12
       max=200 ----+';
  put dy nldatemdl.;
  put dy nldatemdl5.;
  put dy nldatemdl9.;
  put dy nldatemdl12.;
  put dy nldatemdl200.;
run;
```

--- NLDATEMDL min=5 default=12 max=200 ---+
November 19
11/19
Nov 19
November 19
November 19

NLDatemdMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.

Categories: CAS Date

Alignment: Left

Syntax

```
NLDatemdMw.
```

Syntax Description

\[ w \]

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default  9

Range  5–200
Details

NLDATEMDM writes the date in a medium-uniform pattern with abbreviation of the month and the day using numbers and delimiters, such as Nov 19.

Example

The following example specifies the date of November 19, 2012.

```sas
data _null_;
  dt = dt='19Nov2012:00:00:00'dt;
  dy='19Nov2012'd;
  put '++++ NLDATEMDL min=5 default=9 max=200 ++++';
  put dy nldate mdm.;
  put dy nldate mdm5.;
  put dy nldate mdm9.;
  put dy nldate mdm200.;
run;
```

| ++++ NLDATEMDL min=5 default=9 max=200 ++++ |
| Nov 19 |
| 11/19 |
| Nov 19 |
| Nov 19 |

NLDATEMDSw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.

Categories: CAS
Date

Alignment: Left

Syntax

NLDATEMDSw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 5
Range 5–200
Details
NLDATEMDS writes the date in a short-uniform pattern in full length of the month and the day using numbers and delimiters, such as MM/dd.

Example
The following example specifies the date of November 19, 2012.

```sas
data _null_;
  dt = dt='19Nov2012:00:00:00'dt;
  dy='19Nov2012'd;
  put '----- NLDATEMDL min=5 default=5 max=200 -----';
  put dy nldatemds.;
  put dy nldatemds5.;
  put dy nldatemds5.;
  put dy nldatemds200.;
run;
```

```plaintext
----- NLDATEMDL min=5 default=5 max=200 -----
11/19
11/19
11/19
11/19
```

NLDatemDnw. Format
Converts a SAS date value to the date value of the specified locale, and then writes the value as the name of the month.

Categories:
- CAS
- Date

Alignment: Left

Syntax
NLDatemDnw.

Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the name of the month to fit the format width.

- Default: 9
- Range: 4–200
Comparisons

`NLDATEMNw.` is similar to `MONNAMEw.`, except that `NLDATEMNw.` is locale-specific.

Example

These examples specify the input value of 15760, which is the SAS date value that corresponds to February 24, 2003. The first example specifies the `en_US` locale. The second example specifies the `german_Germany` locale.

```sas
option locale=en_US;
data _null_;  
  month=15760;  
  put month nldatemn.;  
run;

February

option locale=german_germany;
data _null_;  
  month=15760;  
  put month nldatemn.;  
run;

Februar
```

See Also

Formats:

- "`NLDATEw. Format` on page 154"
- "`NLDATEWw. Format` on page 163"
- "`NLDATEWNw. Format` on page 165"

### NLDATESw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date string.

**Categories:** CAS  
**Date**  
**Alignment:** Left

**Syntax**

```
NLDATESw.
```
Syntax Description

**w**

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default** 10

**Range** 2–200

Details

NLDATES writes the date in a short-uniform pattern that contains only numbers and delimiters, such as mm/dd/yyyy.

Example

This example specifies the date November 19, 2012. This example specifies the en_US locale.

```sas
option locale=en_US;
data _null_;
  dt = dt='19Nov2012:00:00:00'dt;
  dy='19Nov2012'd;
  put '+--- NLDATEMDL min=2 default=10 max=200 ---+';
  put dy nldates.;
  put dy nldates8.;
  put dy nldates10.;
  put dy nldates200.;
run;
```

```
+--- NLDATEMDL min=2 default=10 max=200 ---+
11/19/2012
11/19/12
11/19/2012
11/19/2012
```

--- NLDATEMDL min=2 default=10 max=200 ----+
11/19/2012
11/19/12
11/19/2012
11/19/2012

**NLDAT**

Converts a SAS date value to the date value of the specified locale, and then writes the value as the date and the day of the week.

**Categories:** CAS

**Date**

**Alignment:** Left

--- NLDATEWw. Format

Syntax

**NLDAT**
Syntax Description

\(w\)

specifies the width of the output field. If necessary, SAS abbreviates the date and the day of the week to fit the format width.

Default 29

Range 10–200

Details

The NLDATEW format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 85 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Comparisons

NLDATEWw. is similar to WEEKDATEw., except that NLDATEWw. is locale specific.

Example

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003. The first example specifies the en_US locale. The second example specifies the de_DE locale.

```sas
options locale=en_US;
data _null_; dy=15760; put dy nldatew.; put dy nldatew20.; put dy nldatew200.; run;
```

Monday, February 24, 2003
Mon, Feb 24, 2003

Monday, February 24, 2003

```sas
options locale=de_DE;
data _null_; dy=15760; put dy nldatew.; put dy nldatew20.; put dy nldatew200.; run;
```

Mo., 24. Feb 2003
Mo., 24. Feb 2003
Montag, 24. Februar 2003
NLDATEWNw. Format

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the day of the week.

Categories: CAS Date

Alignment: Left

Syntax

NLDATEWNw.

Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the day of the week to fit the format width.

Default 9

Range 4–200

Comparisons

NLDATEWNw. is similar to DOWNAMEw., except that NLDATEWNw. is locale-specific.

Example

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003. The first example specifies the en_US locale. The second example specifies the de_DE locale.

```sas
options locale=en_US;
data _null_;  
dy=15760;  
  put dy nldatewn.;  
run;
```

Monday
See Also

Formats:

- "NLDATEw. Format" on page 154
- "NLDATEMNw. Format" on page 161
- "NLDATEWw. Format" on page 163

**NLDATEYMw. Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the name of the month.

Categories:  CAS
             Date

Alignment:  Left

**Syntax**

**NLDATEYMw.**

**Syntax Description**

-w

specifies the width of the output field.

Default  16
Range  6–200

**Details**

If you specify a width of 6, but your data is larger than 6, your output contains asterisks: **d=******. To remove the asterisks, you can use PROC LOCALEDATA. The following example uses PROC LOCALEDATA to write the date without the asterisks:

```sas
PROC LOCALEDATA;
  LOAD SASLOCALE;
  MODIFY key=DATE_YYMM_SHORT_FORMAT value='%b %y' ;
  SAVE REGISTRY / _ALL_ syntax=SAS;
```
Example
This example specifies the spanish_Spain locale option. This example specifies the date of April 4, 2019.

```sas
options locale=spanish_Spain;
data _null_; 
  dy=today();
  put dy nldateym.;
  put dy nldateym12.;
  put dy nldateym200.;
run;
```

```
abril de 2019
abr de 2019
abril de 2019
```

See Also

Format:
- "NLDATEmDw. Format" on page 157

NLDATEYMLw. Format
Converts a SAS date value to the date string of the specified locale, and then writes the month and year.

Categories: CAS  
Date  

Alignment: Left

Syntax

```
NLDATEYMLw.
```

Syntax Description

\( w \)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.
Details

NLDATEYML writes the date in a long-uniform pattern with abbreviations for the month and year, such as April 2019.

Example

This example specifies the date April 4, 2019. This example specifies the en_US locale.

```sas
options locale=en_US;
data _null_;  
  dt = datetime();  
  dy = date();  
  put "++++ NLDATEYML min=5 default=14 max=200 ++++";  
  put dy nldateyml.;  
  put dy nldateyml7.;  
  put dy nldateyml11.;  
  put dy nldateyml14.;  
  put dy nldateyml200.;  
run;
```

```
++++ NLDATEYML min=5 default=14 max=200 ++++
April 2019
04/2019
Apr 2019
April 2019
April 2019
```

NLDATEYMMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date values as the month and year with abbreviations.

Categories: CAS
Date

Alignment: Left

Syntax

NLDATEYMMw.
Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 11

Range 5–200

Details

NLDATEYMS writes the date in a medium-uniform pattern with abbreviations for the month and year, such as Apr 2019.

Example

This example specifies the date April 4, 2019. This example specifies the en_US locale.

```sas
options locale=en_US;
data _null_;
dt = datetime();
dy = date();
put "---- NLDATEYML min=5 default=14 max=200 ----";
put dy nldateymm.;
put dy nldateyymm7.;
put dy nldateyymm11.;
put dy nldateyymm200.;
run;
```

---- NLDATEYML min=5 default=14 max=200 ----
Apr 2019
04/2019
Apr 2019
Apr 2019

--- NLDATEYMSw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date and year.

Categories: CAS

Date

Alignment: Left

Syntax

NLDATEYMSw.
Syntax Description

\( w \)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default** 7  
**Range** 5–200

Details

NLDATEYMS writes the date in a short-uniform pattern with numbers and delimiters such as mm/yyyy.

Example

This example specifies the date April 4, 2019. This example specifies the en_US locale.

```sas
options locale=en_US;
data _null_;   
dt = datetime();
dy = date();
put "+--- NLDATEYMS min=5 default=14 max=200 ----+";
put dy nldateyms.;
put dy nldateyms5.;
put dy nldateyms7.;
put dy nldateyms200.;
run;
```

```plaintext
+--- NLDATEYMS min=5 default=14 max=200 ----+
04/2019
04/19
04/2019
04/2019
```

**NLDATEYQw. Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the quarter.

**Categories:** CAS  
**Date**

**Alignment:** Left

**Syntax**

\[ \text{NLDATEYQw.} \]
Syntax Description

w
  specifies the width of the output field.

  Default  16
  Range    4–200

Details

The NLDATEYQ format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 85 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Example

This example specifies the date April 4, 2019. This example specifies the fr_FR locale option.

options locale=fr_FR;
data _null_;    dy=today();    dt=datetime();    put "---- NLDATEYQ min=4 default=16 max=200 ----+";    put ' 16' +5 dy nldateyq.;    put ' 4' +5 dy nldateyq4.;    put ' 14' +5 dy nldateyq14.;    put ' 32' +5 dy nldateyq32.;    put '200' +5 dy nldateyq200.;run;

---- NLDATEYQ min=4 default=16 max=200 ----+
  16  T2 2019
  4   ****
  14  T2 2019
  32  2e trimestre 2019
 200  2e trimestre 2019

NLDATEDYQlw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.

Categories:    CAS
               Date
Alignment:    Left
Syntax

NLDATEYQLw.

Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 18
Range 4–200

Details

NLDATEYQL writes the date in a long-uniform pattern with full length for the year and year’s quarter value, such as 2nd quarter 2019.

Example

This example specifies the date April 4, 2019. This example specifies the en_US locale option.

options locale=en_US;
data _null_
    dt = datetime();
    dy = date();
    put '+--- NLDATEYQL min=4 default=18 max=200 ---+';
    put dy nldateyql.;
    put dy nldateyql6.;
    put dy nldateyql7.;
    put dy nldateyql18.;
    put dy nldateyql200.;
run;

+--- NLDATEYQL min=4 default=18 max=200 ---+
2nd quarter 2019
2019/2
Q2 2019
2nd quarter 2019
2nd quarter 2019

NLDATEYQMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.

Categories: CAS
Date

Alignment: Left
NLDATEYQMw.

Syntax Description

w specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 7

Range 4–200

Details

NLDATEYQM writes the date in a medium-uniform pattern with abbreviations for the year and year’s quarter value, such as Q2 2019.

Example

The following example uses the date April 4, 2019.

```sas
data _null_
    dt = datetime();
    dy = date();
    put '----- NLDATEYQM min=4 default=7 max=200 +++';
    put dy nldateyqm.;
    put dy nldateyqm6.;
    put dy nldateyqm7.;
    put dy nldateyqm200.;
run;
```

```
----- NLDATEYQM min=4 default=7 max=200 +++
Q2 2019
2019/2
Q2 2019
Q2 2019
```

NLDATEYQS w. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (1–4) with numbers and delimiters.

Categories: CAS
Date

Alignment: Left

Syntax

NLDATEYQS w.
Syntax Description

\[ w \]

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default: 6

Range: 4–200

Details

NLDATEYQS writes the date in a short-uniform pattern with numbers and delimiters for the year and year's quarter value, such as 2019/2.

Example

The following example specifies the date April 4, 2019.

```
data _null_;  
dt = datetime();  
dy = date();  
put '+--- NLDATEYQS min=4 default=6 max=200 ---+';  
put dy nldateyqs.;  
put dy nldateyqs4.;  
put dy nldateyqs6.;  
put dy nldateyqs200.;  
run;  
```

```
+--- NLDATEYQS min=4 default=6 max=200 ---+
2019/2
19/2
2019/2
2019/2
```

**NLDATEYRw. Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year.

Categories: CAS Date

Alignment: Left

Syntax

**NLDATEYRw.**
Syntax Description

\texttt{w}

specifies the width of the output field.

Default \hspace{1cm} 16

Range \hspace{1cm} 2–200

Example

This example specifies the fr\_FR locale option. This example specifies the date April 4, 2019.

\begin{verbatim}
options locale=fr_FR;
data _null_;  
dt = datetime();  
dy = date();  
put '--- NLDATEYR min=4 default=6 max=200 ---';  
put dy nldateyr.;  
put dy nldateyr2.;  
put dy nldateyr8.;  
put dy nldateyr200.;  
run;
\end{verbatim}

--- NLDATEYR min=4 default=6 max=200 ---
2019
19
2019
2019

\section*{NLDATEYWw. Format}

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the week.

Categories: CAS \\
Date \\
Alignment: Left

Syntax

\texttt{NLDATEYWw.}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default \hspace{1cm} 16
Example

This example specifies the date April 4, 2019. This example specifies the fr_FR locale option.

```sas
options locale=fr_FR;
data _null_;  
  dt = datetime();  
  dy = date();  
  put "+--- NLDATyw min=5 default=16 max=200 ---+";  
  put '16' +5 dy nldateyw.;  
  put '5' +5 dy nldateyw5.;  
  put '8' +5 dy nldateyw8.;  
  put '32' +5 dy nldateyw32.;  
  put '200' +5 dy nldateyw200.;  
run;
```

```sas
+--- NLDATyw min=5 default=16 max=200 ---+
  16  Week 14 2019
  5  *****
  8  W14 19
  32  Week 14 2019
  200  Week 14 2019
```

NLDATMw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.

Categories:  
CAS  
Date and Time

Alignment:  
Left

Syntax

NLDATMw.

Syntax Description

w  
specifies the width of the output field. If necessary, SAS abbreviates the datetime value to fit the format width.

Default  
30

Range  
10–200
Comparison

The NLDATMw. format is similar to the DATETIMEw. format, except that the NLDATMw. format is locale-specific.

Example

These examples specify the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003. The first example specifies the en_US locale. The second example specifies the de_DE locale.

```sas
options locale=en_US;
data _null_;  
dy=1361709583;  
put dy nldatm.;  
run;

24Feb2003:12:39:43
```

```sas
options locale=de_DE;
data _null_;  
dy=1361709583;  
put dy nldatm.;  
run;

24. Februar 2003 12.39 Uhr
```

See Also

 Formats:
- "NLDATMAPw. Format" on page 177
- "NLDATMTMw. Format" on page 187
- "NLDATMWw. Format" on page 189

NLDATMAPw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime with a.m. or p.m.

Categories: CAS
            Date and Time

Alignment: Left

Syntax

`NLDATMAPw.`
Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the date-time value to fit the format width.

Default 32
Range 16–200

Comparisons

The NLDATMAPw. format is similar to DATEAMPMw., except that the NLDATMAPw. format is locale-specific.

Example

These examples specify the input value of 1361709583, which is the SAS date-time value that corresponds to 12:39:43 p.m. on February 24, 2003. The first example specifies the en_US locale. The second example specifies the es_MX locale.

```sas
options locale=en_US;
data _null_;
dy=1361709583;
put dy nldatmap.;
run;
```

```
February 24, 2003 12:39:43 PM
```

```sas
options locale=es_MX;
data _null_;
dy=1361709583;
put dy nldatmap200.;
run;
```

```
24 de febrero de 2003 12:39:43 p.m.
```

See Also

Formats:
- “NLDATMw. Format” on page 176
- “NLDATMTMw. Format” on page 187
- “NLDATMWw. Format” on page 189
NLDATMDTw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month, day of the month and year.

Categories: CAS
Date
Date and Time

Alignment: Left

Syntax

NLDATMDTw.

Syntax Description

w
  specifies the width of the output field

Default  18
Range    10-200

Example

This example specifies the value 86400, which corresponds to January 2, 1960. The first example specifies the english_United States locale. The second example specifies the turkish_Turkey locale.

```
options locale=en_US;
data _null_;   // English locale
  x=86400;
  put x nldatmdt.;
run;

January 2, 1960
```

```
options locale=tr_TR;
data _null_;   // Turkish locale
  x=86400;
  put x nldatmdt200.;
run;

2 Ocak 1960
```

See Also

Formats:
NLDATMLw. Format

Converts a SAS datetime value to the datetime string of the specified locale in the long representation of the date.

Categories: CAS
Date and Time

Alignment: Left

Syntax

NLDATMLw.

Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 30
Range 9–200

Details

NLDATML writes the date in a long-uniform pattern with the full length of the month, date, year, and time, such as April 10, 2019 03:13:27 PM.

Example

data _null_
  dt = datetime();
  dy = date();
  put "**** NLDATML min=9 default=30 max=200 ****"
    put dt nldatml.;
    put dt nldatml19.;
    put dt nldatml24.;
    put dt nldatml30.;
    put dt nldatml200.;
run;

**** NLDATML min=9 default=30 max=200 ****
April 10, 2019 03:13:27 PM
04/10/2019 15:13:27
Apr 10, 2019 03:13:27 PM
April 10, 2019 03:13:27 PM
April 10, 2019 03:13:27 PM
NLDATMMw. Format

Converts a SAS datetime value to the datetime string of the specified locale in the medium representation of the date.

Categories: CAS
            Date and Time

Alignment: Left

Syntax

\[ \text{NLDATMMw.} \]

Syntax Description

\( w \)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 24

Range 9–200

Details

NLDATMM writes the date in a medium-uniform pattern with abbreviations of the month and week names, such as Apr 10, 2019 03:23:34 PM.

Example

```sas
data _null_;  
dt = datetime();  
dy = date();  
put '----- NLDATMM min=9 default=30 max=200 ----+';  
put dt NLDATMM.;  
put dt NLDATMM19.;  
put dt NLDATMM24.;  
put dt NLDATMM30.;  
put dt NLDATMM200.;  
run;
```

```
----- NLDATMM min=9 default=30 max=200 ----+
Apr 10, 2019 03:23:34 PM
04/10/2019 15:23:34
Apr 10, 2019 03:23:34 PM
Apr 10, 2019 03:23:34 PM
Apr 10, 2019 03:23:34 PM
```
NLDATMMDw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month and the day of the month.

Categories: CAS  
Date  
Date and Time  

Alignment: Left

Syntax

NLDATMMDw.

Syntax Description

w
  specifies the width of the output field.

Default: 16
Range: 6–200

Example

This example uses the en_US locale option.

```sas
options locale=en_US;
data _null_;  
x=put(86400,nldatmmd.);  
put x=;  
r
```

x=January 02

See Also

Format:

  "NLDATMYMw. Format" on page 193

NLDATMMDLw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the full-length of the month and day of the month.
Syntax

**NLDATMMDL**\(_w\).

Syntax Description

\(w\)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default**

12

**Range**

5–200

Details

NLDATMMDL writes the date in a long-uniform pattern with full-length of the month and the day, such as November 19.

Example

The following example uses the date of April 12, 2019.

```sas
data _null_
  dt = datetime()
  dy = date()
  put '+---- NLDATMMDL min=5 default=12 max=200 ----+';
  put dt nldatmmdl.;
  put dt nldatmmdl5.;
  put dt nldatmmdl9.;
  put dt nldatmmdl12.;
  put dt nldatmmdl200.;
run;
```

```
+---- NLDATMMDL min=5 default=12 max=200 ----+
April 12
04/12
Apr 12
April 12
April 12
```

**NLDATMMDMw. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using abbreviations.
**Syntax**

**NLDATMMDMw.**

**Syntax Description**

*w*

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default** 9

**Range** 5–200

**Details**

NLDATMMDM writes the date in a medium-uniform pattern with abbreviations of the month and the day, such as Nov 19.

**Example**

The following example uses the date of April 12, 2019.

```sas
data _null_;  
dt = datetime();  
dy = date();  
put '+--- NLDATMMDM min=5 default=9 max=200 ---+';  
put dt nldatmmdm.;  
put dt nldatmmdm5.;  
put dt nldatmmdm9.;  
put dt nldatmmdm200.;  
run;
```

```plaintext
+--- NLDATMMDM min=5 default=9 max=200 ---+
Apr 12
04/12
Apr 12
Apr 12
```

**NLDATMMDSw. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using numbers and delimiters.
Syntax

NLDATMMDS\(w\).

Syntax Description

\(w\)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 5

Range 5–200

Details

NLDATMMDS writes the date in a short-uniform pattern with numbers and delimiters of the month and the day, such as 4/12.

Example

The following example uses the date of April 12, 2019.

```
data _null_;  
  dt = datetime();  
  dy = date();  
  put '+--- NLDATMMDS min=5 default=5 max=200 ---+';  
  put dt nldatmmds.;  
  put dt nldatmmds5.;  
  put dt nldatmmds5.;  
  put dt nldatmmds200.;  
  run;  
```

```
+--- NLDATMMDS min=5 default=5 max=200 ---+
04/12
04/12
04/12
04/12
04/12
```

NLDATMMN\(w\). Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.

Categories:

CAS
Date
Date and Time
**NLDATMMN_{w}**

**Syntax Description**

- **{w}** specifies the width of the output field.
- **Default:** 9
- **Range:** 4–200

**Example**

This example uses the en_US locale option.

```sas
options locale=en_US;
data _null_;   
  dt = datetime();
  dy = date();
  put '+--- NLDATMMN min=5 default=10 max=200 ---+';
  put dt nldatmmn.;
  put dt nldatmmn4.;
  put dt nldatmmn10.;
  put dt nldatmmn200.;
run;
```

```
+--- NLDATMMN min=5 default=10 max=200 ---+
April
Apr
April
April
```

**NLDATMS_{w}. Format**

Converts a SAS datetime value to the datetime string of the specified locale in the short representation of the date.

**Categories:**
- CAS
- Date and Time

**Syntax**

```sas
NLDATMS_{w}.
```
Syntax Description

\textbf{w}

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 19

Range 9–200

Details

NLDATMS writes the date in a short-uniform pattern with number and delimiters, such as MM/DD/YYYY hh:mm:ss.

Example

This example uses the date April 12, 2019.

```sas
data _null_
  dt = datetime();
  dy = date();
  put '+--- NLDATMS min=9 default=19 max=200---+';
  put dt nldatms.;
  put dt nldatms10.;
  put dt nldatms19.;
  put dt nldatms200.;
run;
```

```
+--- NLDATMS min=9 default=19 max=200---+
04/12/2019 11:50:53
041219 11
04/12/2019 11:50:53
04/12/2019 11:50:53
```

\section*{NLDATMTMw. Format}

Converts the time portion of a SAS datetime value to the time-of-day value of the specified locale, and then writes the value as a time of day.

Categories:
- CAS Date and Time
  - Time

Alignment: Left

Syntax

\textbf{NLDATMTMw.}
Syntax Description

\( w \)

specifies the width of the output field.

Default 16

Range 16–200

Comparisons

The NLDATMTMw. format is similar to the TODw. format, except that the NLDATMTMw. format is locale-specific.

Example

These examples use the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003.

This example specifies the English_United_States locale.

```plaintext
options locale=en_US;
data one;
   event=1361709583;
   put event nldatmtm.;
run;
```

12:39:43

This example specifies the German_Germany locale.

```plaintext
options locale=en_US;
data one;
   event=1361709583;
   put event nldatmtm.;
run;
```

12.39 Uhr

See Also

Formats:
- “NLDATMw. Format” on page 176
- “NLDATMAPw. Format” on page 177
- “NLDATMWw. Format” on page 189
NLDATMTZw. Format

Converts the time portion of the SAS datetime value to the time of day and time zone of the specified locale.

Categories: CAS
            Date and Time
            Time

Alignment: Left

Syntax

NLDATMTZw.

Syntax Description

w
  specifies the width of the output field.

  Default  32
  Range    16–200

Example

This example uses the current datetime value. This example specifies the French_France locale.

    options locale=fr_FR;
    data test;
      x=datetime();
      put x=nldatmtz.;
    run;

  x=10 h 08 -0400

NLDATMWw. Format

Converts SAS datetime values to the locale sensitive datetime string as the day of the week and the datetime.

Categories: CAS
            Date and Time

Alignment: Left
Syntax

NLDATMWw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the day of week and datetime to fit the format width.

Default 41

Range 16–200

Details

The NLDATMW format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 85 for information about recommended widths for locale and encoding combinations. You might need to use the suggested width for the NLDATMW format.

Comparisons

The NLDATMWw. format is similar to the TWMDYw. format, except that the NLDATMWw. format is locale-specific.

Example

This example uses the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003. This example specifies the English_United States locale.

options locale=English_UnitedStates;
data one;
x=put(1361709583,nldatmw.);
y=put(1361709583,nldatmw30.);
z=put(1361709583,nldatmw200.);
put x=;
put y=;
put z=;
run;

x=Monday, February 24, 2003 12:39:43 PM
y=Mon, Feb 24, 2003 12:39:43 PM
z=Monday, February 24, 2003 12:39:43 PM

See Also

Formats:

- “NLDATMw. Format” on page 176
NLDATMWNw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as the day of the week.

Categories: CAS
            Date and Time

Alignment: Left

Syntax

\texttt{NLDATMWNw}.

Syntax Description

\textbf{w}

\begin{itemize}
  \item specifies the width of the output field.
\end{itemize}

\begin{center}
\begin{tabular}{l|l}
  Default & 9 \\
  Range   & 4–200
\end{tabular}
\end{center}

Example

This example writes the SAS datetime value as a day of the week. This example specifies the English United States locale, which is the default SAS session encoding.

\begin{verbatim}
data _null_;  
  dt = datetime();  
  dy = date();  
  put '+' NLDATMWN min=4 default=9 max=200 ----';  
  put dt nldatmwn;  
  put dt nldatmwn4.;  
  put dt nldatmwn9.;  
  put dt nldatmwn200.;  
run;
\end{verbatim}

\begin{verbatim}
+----- NLDATMWN min=4 default=9 max=200 ----
Tuesday
Tue
Tuesday
Tuesday
\end{verbatim}
NLDATMWZw. Format

Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.

Categories: CAS Date Date and Time

Alignment: Left

Syntax

NLDATMWZw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the day of week and datetime to fit the format width.

Default: 40

Range: 16–200

Details

The NLDATMWZ format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 85 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Example

This example uses the current datetime value. This example specifies the French_France locale.

```sas
options locale=fr_FR;
data test;
x=datetime();
put x=nldatmwz.;
run;
```

x=mardi 16 avril 2019 10:35:06 -0400
NLDATMYMw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the month and year.

**Categories:**
- CAS
- Date and Time

**Syntax**

```
NLDATMYMw.
```

**Syntax Description**

- `w`
  - specifies the width of the output field.
  - Default: 16
  - Range: 6–200

**Example**

This example uses the `en_US` locale option.

```sas
options locale=en_US;
data _null_;  
x=put(86400,nldatym.);  
y=put(86400,nldatym12.);  
put x=;  
put y=;  
run;
```

```
x=January 1960  
y=January 1960
```

**See Also**

**Format:**

- "NLDATMMDw. Format" on page 182
**NLDATMYMLw. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.

**Categories:**
- CAS
- Date

**Alignment:**
- Left

**Syntax**

`NLDATMYMLw.`

**Syntax Description**

`w`

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 14
- **Range:** 5–200

**Details**

`NLDATMYML` writes the date in a long-uniform pattern with full length of the month and year, such as November 2012.

**Example**

The following example uses the date April 16, 2019.

```sas
data _null_
  dt = datetime();
  put '++++ NLDATMYML min=5 default=14 max=200 ++++
      April 2019
    04/2019
    Apr 2019
    April 2019
    April 2019';
  put dt nldatmyml.;
  put dt nldatmyml7.;
  put dt nldatmyml1l.;
  put dt nldatmyml14.;
  put dt nldatmyml200.;
run;
```

```
++++ NLDATMYML min=5 default=14 max=200 ++++
April 2019
04/2019
Apr 2019
April 2019
April 2019
```
NLDATMYMMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.

Categories:
- CAS
- Date

Alignment: Left

Syntax

**NLDATMYMM**w.

Syntax Description

**w**

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>5–200</td>
</tr>
</tbody>
</table>

Details

NLDATMYMM writes the date in a medium-uniform pattern with abbreviations of the month and year, such as Nov 2012.

Example

The following example uses the date April 16, 2019.

```sas
data _null_
  dt = datetime();
  dy = date();
  put '++++ NLDATMYMM min=5 default=11 max=200 ++++';
  put dt nldatmymm.;
  put dt nldatmymm7.;
  put dt nldatmymm11.;
  put dt nldatmymm200.;
run;
```

```
++++ NLDATMYMM min=5 default=11 max=200 ++++
Apr 2019
04/2019
Apr 2019
Apr 2019
```
NLDATMYMSw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the month and year with numbers and a delimiter.

Categories:
CAS
Date

Alignment:
Left

Syntax

NLDATMYMSw.

Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default
7

Range
5–200

Details

NLDATMYMS writes the date in a short-uniform pattern with numbers and a delimiter for the month and year, such as 11/2012.

Example

The following example uses the date April 16, 2019.

```
data _null_;  
  dt = datetime();  
  dy = date();  
  put '++++ NLDATMYMS min=5 default=7 max=200 ++++';  
  put dt nldatmyms.;  
  put dt nldatmyms5.;  
  put dt nldatmyms7.;  
  put dt nldatmyms200.;  
run;
```

++++ NLDATMYMS min=5 default=7 max=200 ++++
04/2019
04/19
04/2019
04/2019
NLDATMYQw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the quarter and the year.

Categories:
CAS
Date and Time

Alignment:
Left

Syntax

NLDATMYQw.

Syntax Description

w
specifies the width of the output field.

Default
16

Range
4–200

Details

The NLDATMYQ format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 85 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Example

This example uses the en_US locale option.

```sas
options locale=en_US;
data _null_;
    dy=today();
    dt=datetime();
    put "+--- NLDATMYQ min=4 default=16 max=200 ---+";
    put ' 16' +5 dt nldatmyq.;
    put '  4' +5 dt nldatmyq4.;
    put ' 14' +5 dt nldatmyq14.;
    put ' 32' +5 dt nldatmyq32.;
    put '200' +5 dt nldatmyq200.;
run;
```
NLDATMYQLw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter value (1–4) and the year.

Category: Date

Alignment: Left

Syntax

NLDATMYQLw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 18

Range 4–200

Details

NLDATMYQL writes the date in a long uniform pattern in full length of the year’s quarter and then the year, such as 4th quarter 2012.

Example

The following example uses the date of April 16, 2019. This example specifies the en_US locale.

```sas
options locale=en_US;
data _null_;  
  dt = datetime();  
  dy = date();  
  put '+' NLDATMYQL min=4 default=18 max=200 '+';
  put dt nldatmyql.;  
  put dt nldatmyql6.;  
  put dt nldatmyql7.;  
  put dt nldatmyql8.;  
  put dt nldatmyql200.;  
run;
```
NLDATMYQMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter (1–4) and then the year.

Categories: CAS
            Date
Alignment: Left

Syntax

NLDATMYQMw.

Syntax Description

w
  specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

  Default  7
  Range    4–200

Details

NLDATMYQM writes the date in a medium-uniform pattern of the year’s quarter and then the year, such as Q2 2019.

Example

The following example uses the date of April 16, 2019.

data _null_
  dt = datetime();
  dy = date();
  put '---- NLDATMYQM min=4 default=7 max=200 ----+';
  put dt nldatmyqm.;
  put dt nldatmyqm6.;
  put dt nldatmyqm7.;
  put dt nldatmyqm200.;
run;
NLDATMYQS. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the quarter (1-4) using numbers and a delimiter.

Categories: CAS
            Date

Alignment: Left

Syntax

NLDATMYQS

Syntax Description

\[ w \]

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 6

Range 4–200

Details

NLDATMYQS writes the date in a short-uniform pattern of the year and year's quarter value using numbers and a delimiter, such as 2012/4.

Example

The following example uses the date April 16, 2019.

data _null_
  dt = datetime();
  dy = date();
  put '--- NLDATMYQM min=4 default=7 max=200 ---+
Q2 2019
2019/2
Q2 2019
Q2 2019';
put dt nldatmyqs.;
put dt nldatmyqs4.;
put dt nldatmyqs6.;
put dt nldatmyqs200.;
run;
NLDATMYRw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.

Categories: CAS
            Date and Time

Alignment: Left

Syntax

NLDATMYRw.

Syntax Description

w

  specifies the width of the output field.

  Default  16
  Range    2–200

Example

This example uses the en_US locale option, which is the default SAS session encoding.

data _null_
  dt = datetime();
  dy = date();
  put '---- NLDATMYR min=2 default=16 max=200 ----+';
  put dt nldatmyr. ;
  put dt nldatmyr2. ;
  put dt nldatmyr32. ;
  put dt nldatmyr200. ;
run;

---- NLDATMYR min=2 default=16 max=200 ----+
2019
19
2019
2019
NLDATMYWw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the week number and the year.

Categories: CAS
Date and Time
Alignment: Left

Syntax

NLDATMYWw.

Syntax Description

w

specifies the width of the output field.

Default 16

Range 5–200

Example

This example uses the en_US locale option, which is the default SAS session encoding.

```
data _null_
  dt = datetime()
  dy = date()
  put '+--- NLDATMYW min=5 default=16 max=200 ---+';
  put ' 16' +5 dt nldatmyw.;
  put '  5' +5 dt nldatmyw5.;
  put '  8' +5 dt nldatmyw8.;
  put ' 32' +5 dt nldatmyw32.;
  put '200' +5 dt nldatmyw200.;
run;
```

```
+--- NLDATMYW min=5 default=16 max=200 ---+
  16     Week 16 2019
  5     *****
  8     W16 19
 32     Week 16 2019
200     Week 16 2019
```
## NLDATMZw Format

Converts SAS datetime values to the locale-sensitive datetime string as datetime and time zone.

### Categories:
- CAS
- Date and Time

### Alignment:
- Left

### Syntax

```
NLDATMZw.
```

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 40
  - Range: 16–200

### Example

This example uses the current datetime value. This example specifies the fr_FR locale.

```
options locale=fr_FR;
data test;
x=datetime();
put x=nldatmz.;
run;
```

```
x=16 avril 2019 11 h 40 -0400
```

## NLMNIAEDw.d Format

Writes the monetary format of the international expression for the United Arab Emirates.

### Categories:
- CAS
- Numeric

### Alignment:
- Left

### Syntax

```
NLMNIAEDw.d
```

Syntax Description

w
specifies the width of the output field.
   Default 12
   Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.
   Default 3
   Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmniaed32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AED1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

"NLMNLAEDx.d Format" on page 240

NLMNIAUDw.d Format

 Writes the monetary format of the international expression for Australia.

Categories: CAS
            Numeric

Alignment: Left

Syntax

NLMNIAUDw.d
Syntax Description

**w**

specifies the width of the output field.

Default 12

Range 8–32

**d**

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English UnitedStates.

```plaintext
x=put(-1234.56789, nlmniaud32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AUD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNLAUDw.d Format" on page 241

---

**NLMNIBGNw.d Format**

Writes the monetary format of the international expression for Bulgaria.

**Categories:**
- CAS
- Numeric

**Alignment:**
- Left

**Syntax**

*NLMNIBGNw.d*
Syntax Description

**w**
- specifies the width of the output field.
  - Default: 12
  - Range: 8–32

**d**
- specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmibgn32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BGN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- Format:
  - "NLMNLBGNw.d Format" on page 242

NLMNIBRLw.d Format

Writes the monetary format of the international expression for Brazil.

Categories: CAS
- Numeric

Alignment: Left

Syntax

NLMNIBRL*w.d
Syntax Description

\( w \)

specifies the width of the output field.

- Default: 12
- Range: 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
  x=put(-1234.56789,nlmnbrl32.2);
  y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BRL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- Format:
  - "\texttt{NLMNLBRLw.d Format} on page 243"

\textbf{NLMNICADw.d Format}

Writes the monetary format of the international expression for Canada.

- Categories: CAS, Numeric
- Alignment: Left

\textbf{Syntax}

\begin{verbatim}
NLMNICADw.d
\end{verbatim}
Syntax Description

\( w \)

specifies the width of the output field.

Default: 12
Range: 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789,\text{nlmnicad32.2}); \]
\[ y=\text{put}(-1234.56789,\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CAD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLCADw.d Format" on page 244

NLMNICHFw.d Format

Writes the monetary format of the international expression for Liechtenstein and Switzerland.

Categories: CAS, Numeric
Alignment: Left

Syntax

NLMNICHFw.d
Syntax Description

\texttt{w}

specifies the width of the output field.

Default \hspace{1cm} 12

Range \hspace{1cm} 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default \hspace{1cm} 2

Range \hspace{1cm} 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmichf32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CHF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- \texttt{"NLMNCHFw.d Format" on page 245}

\textbf{NLMNICNYw.d Format}

Writes the monetary format of the international expression for China.

Categories: \hspace{1cm} CAS

\hspace{2cm} Numeric

Alignment: \hspace{1cm} Left

Syntax

\texttt{NLMN\textsc{icn}Yw.d}
Syntax Description

\( w \)

specifies the width of the output field.

Default: 12

Range: 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 02

Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

\[
\begin{align*}
\text{x=} & \text{put (-1234.56789, nlmnicny32.2);} \\
\text{y=} & \text{put (-1234.56789, dollar32.2)};
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CNY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLCNYw.d Format" on page 246

NLMNICZKw.d Format

Writes the monetary format of the international expression for the Czech Republic.

Categories:

- CAS
- Numeric

Alignment:

Left

Syntax

\[
\text{NLMNICZK}w.d
\]
Syntax Description

\( w \)

specifies the width of the output field.

Default: 12
Range: 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 4
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789,\text{nlmnczk}32.2); \]
\[ y=\text{put}(-1234.56789,\text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x=; )</td>
<td>(CZK1,234.57)</td>
</tr>
<tr>
<td>put ( y=; )</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLCZKw.d Format" on page 247

NLMNIDKKw.d Format

Writes the monetary format of the international expression for Denmark, Faroe Island, and Greenland.

Categories: CAS, Numeric

Alignment: Left

Syntax

NLMNIDKKw.d
Syntax Description

w
specifies the width of the output field.
Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.
Default 2
Range 0–28

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnidkk32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(DKK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLMNLDKKw.d Format" on page 248

NLMNIEEKw.d Format
Writes the monetary format of the international expression for Estonia.

Categories: CAS
Numeric

Alignment: Left

Syntax

NLMNIEEKw.d
Syntax Description

\( w \)

specifies the width of the output field.

Default 12
Range 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default 4
Range 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789, nlmnieek32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EEK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- **Format:**
  - "NLMNLEEkw.d Format" on page 249

---

**NLMNIEGPw.d Format**

Writes the monetary format of the international expression for Egypt.

Categories: CAS
Numeric
Alignment: Left

Syntax

NLMNIEGPw.d
Syntax Description

\(w\)

specifies the width of the output field.

Default: 12
Range: 8–32

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 3
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[x=\text{put}(-1234.56789,\text{nlmniegp32.2});\]
\[y=\text{put}(-1234.56789,\text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put} x=;</td>
<td>\text{(EGP1,234.57)}</td>
</tr>
<tr>
<td>\text{put} y=;</td>
<td>\text{$-1,234.57$}</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLEG\(Pw.d\) Format" on page 250

NLMNIEUR\(w.d\) Format

Writes the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

Categories: CAS
Numeric

Alignment: Left
Syntax

**NLMNIEURw.d**

Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `Locale=German_Germany`.

```plaintext
x=put(-1234.56789,nlmnieur32.2);
y=put(-1234.56789,nlmnieur32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>x=-1.234,57 EUR</td>
</tr>
<tr>
<td>put y=;</td>
<td>y=-1.234,57 €</td>
</tr>
</tbody>
</table>

See Also

Format:
- “**NLMNLEURw.d Format**” on page 251

---

**NLMNIGBPw.d Format**

Writes the monetary format of the international expression for the United Kingdom.

Categories: CAS, Numeric

Alignment: Left
Syntax

\[ \text{NLMNIGBP}w.d \]

Syntax Description

\( w \)

specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default**: 2
- **Range**: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{put}(-1234.56789, \text{nlmnigbp}32.2); \\
y=\text{put}(-1234.56789, \text{dollar}32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(GBP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNIGBPw.d Format" on page 252

---

**NLMNIHKDW.d Format**

Writes the monetary format of the international expression for Hong Kong.

**Categories:** CAS

- Numeric

**Alignment:** Left
Syntax

**NLMNIHKDw.d**

Syntax Description

- **w**: specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**: specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnihkd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HKD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNLHKDw.d Format" on page 253

---

**NLMNIHRKw.d Format**

Writes the monetary format of the international expression for Croatia.

**Categories:** CAS, Numeric

**Alignment:** Left
Syntax

**NLMNIHRK**\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default 12

Range  8–32

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2

Range  0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[\begin{align*}
x &= \text{put}(-1234.56789, \text{n}lmn\text{ih}rk32.2); \\
y &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*}\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put }x=; \quad</td>
<td>(HRK1,234.57)</td>
</tr>
<tr>
<td>\text{put }y=; \quad</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "**NLMNLHRK**\(w.d\) Format" on page 254

**NLMNIHUF**\(w.d\) Format

Writes the monetary format of the international expression for Hungary.

Categories:  CAS  
Numeric

Alignment:  Left
Syntax

NLMNIHUF<w,d>

Syntax Description

<\text{w}> specifies the width of the output field.

Default: 12
Range: 8–32

<\text{d}> specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\text{x=put(-1234.56789,nlmnihuf32.2)};
\text{y=put(-1234.56789,dollar32.2)};

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HUF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNLHUF<\text{w,d}> Format” on page 255
Syntax

NLMNIIDRw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmniidr32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(IDR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLIDRw.d Format" on page 256

NLMNIILSw.d Format

 Writes the monetary format of the international expression for Israel.

Categories: CAS
Numeric

Alignment: Left
Syntax

**NLMNIILS**\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 4

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmniils32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ILS1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNLILS\textit{w.d Format} on page 257"
Syntax

`NLMNIINRw.d`

Syntax Description

`w`

specifies the width of the output field.

Default 12

Range 8–32

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniinr32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(INR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLINRw.d Format" on page 258

---

`NLMNIJPYw.d Format`

Writes the monetary format of the international expression for Japan.

Category: Numeric

Alignment: Left
Syntax

NLMNIJPYw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 0
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnijpy32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(JPY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLJPYw.d Format" on page 259

NLMNIKRWw.d Format

Writes the monetary format of the international expression for South Korea.

Categories: CAS
Numeric

Alignment: Left
Syntax

\texttt{NLMNIKRWw.d}

Syntax Description

\textbf{\textit{w}} specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>8–32</td>
</tr>
</tbody>
</table>

\textbf{\textit{d}} specifies the number of digits to the right of the decimal point in the numeric value.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0–28</td>
</tr>
</tbody>
</table>

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnikrw32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
put \texttt{x=};& (KRW1,234.57) \\
put \texttt{y=};& $-1,234.57$ \\
\hline
\end{tabular}

See Also

\textbf{Format:}

- "NLMNLKRWw.d Format" on page 260

\textbf{NLMNILTLw.d Format}

Writes the monetary format of the international expression for Lithuania.

Categories: CAS

Numeric

Alignment: Left
Syntax

NLMNILTLw.d

Syntax Description

w
  specifies the width of the output field.
  Default 12
  Range 8–32

d
  specifies the number of digits to the right of the decimal point in the numeric value.
  Default 4
  Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmniltl32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LTL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
  - "NLMNILTLw.d Format" on page 261

NLMNILVLw.d Format

Writes the monetary format of the international expression for Latvia.

Categories: CAS
            Numeric

Alignment: Left
Syntax

NLMNILVLw.d

Syntax Description

w
specifies the width of the output field.
Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.
Default 4
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnilvl32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LVL1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

“NLMNLLVLw.d Format” on page 262

NLMNIMOPw.d Format

Writes the monetary format of the international expression for Macau.

Categories: CAS
Numeric

Alignment: Left
Syntax

**NLMNIMOP**

Syntax Description

**w**

specifies the width of the output field.

Default: 12

Range: 8–32

**d**

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnimop32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MOP1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNLMOPw.d Format" on page 263

---

**NLMNIMXNw.d Format**

Writes the monetary format of the international expression for Mexico.

Categories: CAS, Numeric

Alignment: Left
Syntax
NLMNIMXNw.d

Syntax Description

\( w \)
specifies the width of the output field.

Default: 12
Range: 8–32

\( d \)
specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2
Range: 0–28

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

\( x = \text{put}(-1234.56789, \text{nlnmixn32.2}); \)
\( y = \text{put}(-1234.56789, \text{dollar32.2}); \)

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MXN1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNLMXNw.d Format” on page 264

NLMNIMMYRw.d Format

Writes the monetary format of the international expression for Malaysia.

Categories:
- CAS
- Numeric

Alignment: Left
Syntax

**NLMNIMYRw.d**

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnimy32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MYR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- ["NLMNLMYRw.d Format" on page 265](#)

---

**NLMNINOKw.d Format**

 Writes the monetary format of the international expression for Norway.

<table>
<thead>
<tr>
<th>Categories:</th>
<th>CAS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Numeric</td>
</tr>
</tbody>
</table>

| Alignment: | Left |
Syntax

NLMNINOK\texttt{w.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 12

Range 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\texttt{x=put(-1234.56789,nlmninok32.2);}  
\texttt{y=put(-1234.56789,dollar32.2);}  

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NOK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNINOK\texttt{w.d Format}" on page 266

NLMNINZD\texttt{w.d Format}

Writes the monetary format of the international expression for New Zealand.

Categories: CAS Numeric

Alignment: Left
Syntax

\texttt{NLMNINZDw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmninzd32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{tabular}{ll}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=; } & \texttt{(NZD1,234.57)} \\
\texttt{put y=; } & \texttt{\$-1,234.57} \\
\hline
\end{tabular}

See Also

\textbf{Format:}

- "\texttt{NLMNLNZDw.d Format} on page 267"

\section*{NLMNIPLNw.d Format}

Writes the monetary format of the international expression for Poland.

\begin{itemize}
\item \textbf{Categories:} \texttt{CAS}
\item \texttt{Numeric}
\item \texttt{Left}
\end{itemize}
Syntax
NLMNIPLNw.d

Syntax Description

w
specifies the width of the output field.
Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.
Default 2
Range 0–28

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnipln32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(PLN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNIRUBw.d Format” on page 268

NLMNIRUBw.d Format
Writes the monetary format of the international expression for Russia.

Categories: CAS
Numeric

Alignment: Left
Syntax

`NLMNIRUBw.d`

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmnirub32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(RUB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- “`NLMNRUBw.d Format`” on page 269
Syntax

\texttt{NLMNISEKw.d}

Syntax Description

\( w \)

specifies the width of the output field.

- Default: 12
- Range: \( 8 \text{–} 32 \)

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: \( 0 \text{–} 28 \)

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnisek32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SEK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "\texttt{NLMNLSEKw.d Format}" on page 270

\texttt{NLMNISGDw.d Format}

Writes the monetary format of the international expression for Singapore.

Categories: CAS
Numeric

Alignment: Left
Syntax

**NLMNISGDw.d**

Syntax Description

**w**

specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

**d**

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default**: 2
- **Range**: 0–28

Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```plaintext
x=put(-1234.56789, nlmnisgd32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SGD1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNLSGDw.d Format" on page 271

---

### NLMNITHBw.d Format

Writes the monetary format of the international expression for Thailand.

**Categories:**
- CAS
- Numeric

**Alignment:**
- Left
Syntax

NLMNITHB\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default 12

Range 8–32

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\(x=\text{put}(-1234.56789,\text{nlmnithb}32.2);\)
\(y=\text{put}(-1234.56789,\text{dollar}32.2);\)

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLTHB\(w.d\) Format" on page 272

NLMNITRY\(w.d\) Format

Writes the monetary format of the international expression for Turkey.

Categories: CAS

Numeric

Alignment: Left
Syntax

\texttt{NLMNITRYw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default \hspace{1cm} 12

Range \hspace{1cm} 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default \hspace{1cm} 4

Range \hspace{1cm} 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\texttt{x=put(-1234.56789,nlmnity32.2);}  
\texttt{y=put(-1234.56789,dollar32.2);}  

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TRY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

\textbf{Format:}

\begin{itemize}
\item \texttt{"NLMNLTRYw.d Format" on page 273}
\end{itemize}

\textbf{NLMNITWDw.d Format}

Writes the monetary format of the international expression for Taiwan.

Categories: CAS  
            Numeric

Alignment: Left
Syntax

NLMNITWD\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnitwd32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TWD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLTWD\textit{w.d Format}” on page 274

NLMNIUSD\textit{w.d Format}

Writes the monetary format of the international expression for Puerto Rico and the United States.

Categories:
- CAS
- Numeric

Alignment: Left
Syntax

NLMNIUSD\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmniusd32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{verbatim}
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(USD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>
\end{verbatim}

See Also

\textbf{Format:}

\begin{itemize}
\item "NLMNLU$D\textit{w.d Format}" on page 275
\end{itemize}
Syntax

\texttt{NLMNIZAR w.d}

Syntax Description

\textit{w} specifies the width of the output field.

- Default: 12
- Range: 8–32

\textit{d} specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnizar32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ZAR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- Format: "\texttt{NLMNLZARw.d Format}" on page 276

\textbf{NLMNLAEDx.d Format}

Writes the monetary format of the local expression for the United Arab Emirates.

Categories: CAS
            Numeric

Alignment: Left
Syntax

\textbf{NLMNLAEDw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12

Range: \(8\text{–}32\)

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 3

Range: \(0\text{–}28\)

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English_UnitedStates}.

\begin{verbatim}
  x=put(-1234.56789,nlmnlaed32.2);
  y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AED1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$\text{-}1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- \texttt{"NLMNIAEDw.d Format" on page 203}

\textbf{NLMNLAUDw.d Format}

 Writes the monetary format of the local expression for Australia.

Categories: CAS

Numeric

Alignment: Left
Syntax

NLMNLAUD<w>.d

Syntax Description

<w>

specifies the width of the output field.

Default 12
Range 8–32

<d>

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlaud32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AU$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIAUD<w>.d Format” on page 204

NLMNLBGN<w>.d Format

Writes the monetary format of the local expression for Bulgaria.

Categories: CAS
            Numeric

Alignment: Left
Syntax

**NLMNLBGN**\textit{w.d}

Syntax Description

\textit{w} specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

\textit{d} specifies the number of digits to the right of the decimal point in the numeric value.

- **Default**: 2
- **Range**: 0–28

Example

In the following example, the LOCALE= system option is set to English\_United\_States.

\begin{verbatim}
x=put(-1234.56789,nlmnlbgn32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BGN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "**NLMNIBGN**\textit{w.d Format}" on page 205

---

**NLMNLBRLw.d Format**

Writes the monetary format of the local expression for Brazil.

**Categories:**
- CAS
- Numeric

**Alignment:** Left
Syntax

`NLMNLBRLw.d`

Syntax Description

`w`

specifies the width of the output field.

Default: 12

Range: 8–32

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmnlbrl32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(R$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “`NLMNIBRLw.d Format`” on page 206

---

**NLMNLCADw.d Format**

Writes the monetary format of the local expression for Canada.

Categories: CAS, Numeric

Alignment: Left
Syntax

NLMNLCA Dw.d

Syntax Description

w
specifies the width of the output field.

Default: 12
Range: 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlcad32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CA$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNICADw.d Format" on page 207
Syntax

\texttt{NLMNLCHFw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default \hspace{1em} 12

Range \hspace{1em} 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default \hspace{1em} 2

Range \hspace{1em} 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnlchf32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
put x=; & SFr.1,234.57 \\
put y=; & $-1,234.57 \\
\hline
\end{tabular}
\end{center}

See Also

Format:

- “\texttt{NLMNICHFw.d Format}” on page 208

\section*{NLMNLNCNYw.d Format}

Writes the monetary format of the local expression for China.

Categories: \hspace{1em} CAS

\hspace{1em} Numeric

Alignment: \hspace{1em} Left
Syntax

\texttt{NLMNL\textsubscript{CNY}w.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

- Default: 12
- Range: 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=\text{put(-1234.56789,nlmnlcny32.2)};
y=\text{put(-1234.56789,dollar32.2)};
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\text{put x=} & (RMB1,234.57) \\
\text{put y=} & $-1,234.57$ \\
\hline
\end{tabular}

See Also

\textbf{Format:}

- "\texttt{NLMN\textsubscript{ICNY}w.d Format}" on page 209

\textbf{NLMN\textsubscript{LCZK}w.d Format}

Writes the monetary format of the local expression for the Czech Republic.

\begin{tabular}{|l|l|}
\hline
\textbf{Categories:} & \text{CAS} \\
& \text{Numeric} \\
\textbf{Alignment:} & \text{Left} \\
\hline
\end{tabular}
Syntax

NLMNLCZKw.d

Syntax Description

w
specifies the width of the output field.
Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.
Default 4
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlczk32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CZK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNICZKw.d Format” on page 210

NLMNLDKKw.d Format

Wrtes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.

Categories: CAS
Numeric
Alignment: Left
Syntax

**NLMNLDKK** \(_w,d\)**

Syntax Description

\(w\)

specifies the width of the output field.

**Default**: 12

**Range**: 8–32

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

**Default**: 2

**Range**: 0–28

Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

\[x = \text{put}(-1234.56789, \text{nlmldkk}32.2);\]

\[y = \text{put}(-1234.56789, \text{dollar}32.2);\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format**:

- "**NLMNIDDKKw.d Format**" on page 211

---

**NLMNLEEK** \(_w,d\)** Format

Writes the monetary format of the local expression for Estonia.

**Categories**: CAS, Numeric

**Alignment**: Left
Syntax

**NLMNLEEKw.d**

Syntax Description

- **w** specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d** specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 4
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnleek32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNIEEKw.d Format" on page 212

---

**NLMNLEGEPw.d Format**

Writes the monetary format of the local expression for Egypt.

Categories: CAS, Numeric

Alignment: Left
Syntax

\texttt{NLMNLEG\textit{P}w.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 3

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
  x=put(-1234.56789,nlmnlegp32.2);
  y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EGP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

\textbf{Format:}

- "\texttt{NLMNIEGPw.d Format}" on page 213

\textbf{NLMNLEURw.d Format}

Writes the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

Categories: CAS

Numeric

Alignment: Left
Syntax

`NLMNLEURw.d`

Syntax Description

`w`
- specifies the width of the output field.
- Default: 12
- Range: 8–32

`d`
- specifies the number of digits to the right of the decimal point in the numeric value.
- Default: 2
- Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `German_Germany`.

```plaintext
x=put(-1234.56789,nlmnieur32.2);
y=put(-1234.56789,nlmnieur32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>x=-1.234,57 EUR</td>
</tr>
<tr>
<td>put y=;</td>
<td>y=-1.234,57 €</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNIEURw.d Format” on page 214

NLMNLGBPw.d Format

Writes the monetary format of the local expression for the United Kingdom.

Categories: CAS, Numeric

Alignment: Left
Syntax

NLMNLGBP<sub>w.d</sub>

Syntax Description

\( w \)

specifies the width of the output field.

Default: 12

Range: 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlmnlgbp}32.2); \]
\[ y = \text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(£1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIGBP<sub>w.d</sub> Format” on page 215
Syntax

\texttt{NLMNLHKDw.d}

Syntax Description

\textbf{w}

specifies the width of the output field.

- Default: 12
- Range: 8–32

\textbf{d}

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
  x=put(-1234.56789,nlmnlhkdf32.2);
  y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HK$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- \texttt{“NLMNIHKDw.d Format”} on page 216

\textbf{NLMNLHRKw.d Format}

Writes the monetary format of the local expression for Croatia.

Categories: CAS, Numeric
Syntax

\texttt{NLMNLHRKw.d}

Syntax Description

\textit{w} specifies the width of the output field.

Default: 12
Range: 8–32

\textit{d} specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmnlhrk32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \texttt{x=};</td>
<td>(Kn1,234.57)</td>
</tr>
<tr>
<td>put \texttt{y=};</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

\textbf{Format:}

- \textit{“NLMNIHRKw.d Format” on page 217}

\textbf{NLMNLHUFw.d Format}

Writes the monetary format of the local expression for Hungary.

Categories: CAS, Numeric
Syntax

**NLMNLHUF**\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default \(12\)

Range \(8–32\)

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Default \(2\)

Range \(0–28\)

Example

In the following example, the LOCATE= system option is set to English United States.

\(x=put(-1234.56789,nlmnlhuf32.2)\);

\(y=put(-1234.56789,dollar32.2)\);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Ft1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- “NLMNIHUFw.d Format” on page 218

**NLMNLIDRw.d Format**

Writes the monetary format of the local expression for Indonesia.

Categories: CAS, Numeric
Syntax

**NLMNLIDRw.d**

Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlidr32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Rp1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNIIIDRw.d Format" on page 219

---

**NLMNLILSw.d Format**

Writes the monetary format of the local expression for Israel.

**Categories:**

- CAS
- Numeric
Syntax

NLMNILS\textit{w.d}  

Syntax Description

\textbf{w}

specifies the width of the output field.

Default 12

Range \(8–32\)

\textbf{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 4

Range \(0–28\)

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmnlls32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ILS1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIILSw.d Format” on page 220

NLMNLINR\textit{w.d} Format

Writes the monetary format of the local expression for India.

Categories: CAS, Numeric
Syntax

**NLMNLI**\(\text{N}w.d\)**

Syntax Description

\(w\)  
specifies the width of the output field.  
**Default:** 12  
**Range:** 8–32

\(d\)  
specifies the number of digits to the right of the decimal point in the numeric value.  
**Default:** 2  
**Range:** 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmnlr32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(INR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**
- “NLMNII**N**Rw.d Format” on page 221

---

**NLMNLI**\(\text{N}PYw.d\) Format

Writes the monetary format of the international expression for Japan.

**Categories:**
- CAS
- Numeric
Syntax

**NLMNLJPyw.d**

Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 0
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnljpy32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>$(JPY1,234.57)$</td>
</tr>
<tr>
<td>put y=;</td>
<td>$\text{-1,234.57}$</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- “NLMNIJPyw.d Format” on page 222

**NLMNLRWw.d Format**

Writes the monetary format of the local expression for South Korea.

Categories: CAS, Numeric
Syntax

**NLMNLKRW** *w.d*

**Syntax Description**

**w**

specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

**d**

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default**: 0
- **Range**: 0–28

**Example**

In the following example, the **LOCALE=** system option is set to **English_UntitedStates**.

```plaintext
x=put(-1234.56789, nlmnlkrw32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(KRW1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- "**NLMNIKRWW.d Format**" on page 223

---

**NLMNLLTLw.d Format**

Writes the monetary format of the local expression for Lithuania.

**Categories:**

- CAS
- Numeric
Syntax

NLMNLLLTLw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 4
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlltl32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LT1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNILTLLw.d Format" on page 224

NLMNLLLVLw.d Format

Writes the monetary format of the local expression for Latvia.

Categories: CAS, Numeric
Syntax

**NLMNLLVLw.d**

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 4
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnllvl32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Ls1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "**NLMNILVLw.d Format**" on page 225

---

NLMNLMOOpw.d Format

Writes the monetary format of the local expression for Macau.

Categories:
- CAS
- Numeric
Syntax

\texttt{NLMNLMO\textit{p}w.d}

Syntax Description

\textit{w} specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

\textit{d} specifies the number of digits to the right of the decimal point in the numeric value.

- **Default**: 2
- **Range**: 0–28

Example

In the following example, the \texttt{LOCALE= system option is set to English\_UnitedStates.}

\begin{verbatim}
x=put(-1234.56789,nlmnlmop32.2);
y=put(-1234.56789,\texttt{dollar32.2});
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \texttt{x=};</td>
<td>(P1,234.57)</td>
</tr>
<tr>
<td>put \texttt{y=};</td>
<td>S-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "\texttt{NLMNIMOPw.d Format} on page 226"
Syntax

**NLMNLMXNw.d**

**Syntax Description**

`w` specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

`d` specifies the number of digits to the right of the decimal point in the numeric value.

- **Default**: 2
- **Range**: 0–28

**Example**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmnxn32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MX$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- **Format**: "NLMNIMXNw.d Format" on page 227

---

**NLMNLMYRw.d Format**

Writes the monetary format of the local expression for Malaysia.

**Categories**: CAS, Numeric
Syntax

`NLMNLMYRw.d`

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmnlmyr32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(R1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- **Format**:  
  - “`NLMNIMYRw.d Format`” on page 228

---

**NLMNLNO Kw.d Format**

Writes the monetary format of the local expression for Norway.

Categories: CAS, Numeric
Syntax

**NLMNLNOK**\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default: 12

Range: 8–32

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the LOCALE= system option is set to English _UnitedStates._

```plaintext
x=put(-1234.56789,nlmnlnok32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNINOKw.d Format" on page 229

**NLMNLNZDw.d Format**

Writes the monetary format of the local expression for New Zealand.

Categories:

- CAS
- Numeric
Syntax

\texttt{NLMNLNZDw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_United\_States}.

\begin{verbatim}
x=put(-1234.56789,nlmnlznzd32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{verbatim}
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NZ$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>
\end{verbatim}

See Also

Format:

\begin{itemize}
\item “NLMNINZDw.d Format” on page 230
\end{itemize}

\section*{NLMNLPLNw.d Format}

Writes the monetary format of the local expression for Poland.

Categories:

\begin{itemize}
\item CAS
\item Numeric
\end{itemize}
Syntax

\texttt{NLMNLPLNw.d}

Syntax Description

\textit{w} 

specifies the width of the output field.

Default: 12 

Range: 8–32 

\textit{d} 

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2 

Range: 0–28 

Example

In the following example, the \texttt{locale=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnlpln32.2);
y=put(-1234.56789,dollar32.2)
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \texttt{x=;}</td>
<td>(PLN1,234.57</td>
</tr>
<tr>
<td>put \texttt{y=;}</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

\textbf{Format:}

- "\texttt{NLMNIPLNw.d Format}" on page 231

\textbf{NLMNLRLUBw.d Format}

Writes the monetary format of the local expression for Russia.

Categories: CAS
Syntax

\texttt{NLMNLRUBw.d}

Syntax Description

\textbf{w}

specifies the width of the output field.

- Default: 12
- Range: 8–32

\textbf{d}

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnlrub32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(RUB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- Format:
  - \textit{“NLMNIRUBw.d Format” on page 232}

\textbf{NLMNLSEKw.d Format}

Writes the monetary format of the local expression for Sweden.
Syntax

**NLMNLSGDw.d**

**Syntax Description**

**w**

specifies the width of the output field.

- Default: 12
- Range: 8–32

**d**

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

**Example**

In the following example, the `LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlsed32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- "**NLMNISEKw.d Format**" on page 233

---

**NLMNLSGDw.d Format**

Writes the monetary format of the local expression for Singapore.
Syntax

NLMNLSGDw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default \( \) 12

Range \( 8–32 \)

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default \( \) 2

Range \( 0–28 \)

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789, \text{nlmnlsd}32.2); \]
\[ y=\text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SG$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNISGDw.d Format” on page 234

NLMNLTHBw.d Format

Writes the monetary format of the local expression for Thailand.
Syntax

NLMNLTHBw.d

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlthb32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNITHBw.d Format" on page 235

NLMNLTRYw.d Format

Writes the monetary format of the local expression for Turkey.
Syntax

\texttt{NLMNLTRYw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 12

Range 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 4

Range 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnltry32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=;} & (YTL1,234.57) \\
\texttt{put y=;} & $-1,234.57$ \\
\hline
\end{tabular}

See Also

Format:

\begin{itemize}
\item “\texttt{NLMNITRYw.d Format}” on page 236
\end{itemize}

\textbf{NLMNLTWDw.d Format}

Writes the monetary format of the local expression for Taiwan.
Syntax

\texttt{NLMNLTW\textsubscript{d}d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 12
Range 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the \texttt{LOCALE} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnltd32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NT$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNITWD\textsubscript{d}d Format” on page 237

\textbf{NLMNLUSD\textsubscript{d}d Format}

Writes the monetary format of the local expression for Puerto Rico and the United States.
Syntax

**NLMNLUSDw.d**

Syntax Description

- **w** specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d** specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlusd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(US$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

- **Format:**
  - “NLMNUSDw.d Format” on page 238

---

**NLMNLZARw.d Format**

Writes the monetary format of the local expression for South Africa.
Syntax

NLMLNIZAR\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 12

Range 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmnlzar32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>($1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

\begin{itemize}
  \item "NLNNIZAR\textit{w.d} Format" on page 239
\end{itemize}
Syntax

\texttt{NLMNYw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 9

Range 1–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 0

Range 0–31

Details

The NLMNYw.d informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMNYw.d format writes numeric values by using the currency symbol, the thousands separator, and the decimal separator that is used by the locale.

\textbf{Note:} The NLMNYw.d format does not convert currency format. Therefore, the value of the formatted number should equal the currency of the current locale value.

Comparisons

The NLMNYw.d and NLMNYw.d formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYw.d format uses three-letter international currency codes, such as USD, while NLMNYw.d format uses local currency symbols, such as $.

The NLMNYw.d format is similar to the DOLLARw.d format, except that the NLMNYw.d format is locale-specific.

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English_UnitedStates}.

\begin{verbatim}
  x=put(-1234.56789, nlmny32.2);
  y=put(-1234.56789, dollar32.2);
\end{verbatim}
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>($1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**
- “NLMNYIw.d Format” on page 279

**Informat:**
- “NLMNYw.d Informat” on page 628
- “NLMNYIw.d Informat” on page 629

---

**NLMNYIw.d Format**

Writes the monetary format of the international expression in the specified locale.

**Categories:**
- CAS
- Numeric

**Alignment:**
- Left

**Syntax**

\[ \text{NLMNYI} \wedge .d \]

**Syntax Description**

\[ \wedge \]

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\[ .d \]

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 0
- **Range:** 0–31
Details

The NLMNYIw.d informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMNYIw.d format writes numeric values by using the international currency code, and locale-dependent thousands and decimal separators. The position of international currency code is also locale dependent.

**Note:** The NLMNYIw.d format does not convert currency format. Therefore, the value of the formatted number should equal the currency of the current locale value.

Comparisons

The NLMNYw.d and NLMNYIw.d formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYIw.d format uses three-letter international currency codes, such as USD, while NLMNYw.d format uses local currency symbols, such as $.

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnyi32.2);
y=put(-1234.56789,nlmny32.2);
z=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(USD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>($1,234.57)</td>
</tr>
<tr>
<td>put z=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

**Format:**
- "NLMNYw.d Format" on page 277

**Informats:**
- "NLMNYw.d Informat" on page 628
- "NLMNYIw.d Informat" on page 629
NLNUMw.d Format

Writes the numeric format of the local expression in the specified locale.

Categories: CAS
Numeric

Alignment: Left

Syntax

**NLNUMw.d**

Syntax Description

**w**

specifies the width of the output field.

Default 6

Range 1–32

**d**

specifies to divide the number by $10^d$. If the data contains decimal separators, the $d$ value is ignored.

Default 0

Range 0–31

Details

The NLNUMw.d informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLNUMw.d format writes numeric values by using the thousands separator and the decimal separator that is used by the locale.

Comparisons

The NLNUMw.d format writes the numeric value with locale-dependent thousand and decimal separators. The NLNUMIw.d format writes the numeric value with a comma (,) as thousands separator and a period (.) as a decimal separator.

If the $w$ or $d$ values are not large enough to generate a formatted number, the NLNUMw.d format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

Example

```plaintext
x=put(-1234356.7891,nlnum32.2);
```
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234,356.79</td>
</tr>
<tr>
<td>options LOCALE=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>-1.234.356,79</td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLNUMIw.d Format” on page 282

**Informats:**
- “NLNUMw.d Informat” on page 630
- “NLNUMIw.d Informat” on page 632

### NLNUMIw.d Format

Writes the numeric format of the international expression in the specified locale.

#### Categories:
- CAS
- Numeric

#### Alignment:
- Left

#### Syntax

**NLNUMIw.d**

#### Syntax Description

**w**
- specifies the width of the output field.
  - Default: 6
  - Range: 1–32

**d**
- specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 0
Details
The NLNUMI \(w.d\) informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLNUMI \(w.d\) format writes numeric values by using a comma (,) as thousands separator and a period (.) as a decimal separator for all locales.

Comparisons
The NLNUMI \(w.d\) format writes the numeric data of the international expression in the specified locale. The NLNUMI \(w.d\) format writes the numeric value with a comma (,) as thousands separator and a period (.) as a decimal separator.

If the \(w\) or \(d\) values are not large enough to generate a formatted number, the NLNUMI \(w.d\) format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

Example
\[
x=put(-1234356.7891,nlnumi32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates; put x=;</td>
<td>-1,234,356.79</td>
</tr>
<tr>
<td>options LOCALE=German_Germany; put x=;</td>
<td>-1,234,356.79</td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLNUMIw.d Format" on page 281

Informats:
- "NLNUMIw.d Informat" on page 630
- "NLNUMIw.d Informat" on page 632

NLPCTw.d Format
Writes percentage data of the local expression in the specified locale.
Syntax

**NLPCTw.d**

**Syntax Description**

**w**
- Specifies the width of the output field.
- Default: 6
- Range: 4–32

**d**
- Specifies to divide the number by $10^d$. If the data contains decimal separators, the $d$ value is ignored.
- Default: 0
- Range: 0–31

**Comparisons**

The NLPCTw.d format writes percentage data of the local expression in the specified locale. The NLPCTw.d format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTIw.d format writes the percentage value with a comma (,) as thousands separator and a period (.) as a decimal separator.

The NLPCTw.d format is similar to the PERCENTw.d format except the NLPCTw.d format is locale-specific.

**Example**

```plaintext
x=put(-12.3456789,nlpct32.2);
y=put(-12.3456789,nlpcti32.2);
z=put(-12.3456789,percent32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>( 1234.57%</td>
</tr>
<tr>
<td>put z=;</td>
<td></td>
</tr>
</tbody>
</table>
Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=German_Germany;</td>
<td>-1.234,57%</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>(1234.57%)</td>
</tr>
<tr>
<td>put z=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

**Format:**
- "NLPCTIw.d Format" on page 285

**Informats:**
- "NLPCTw.d Informat" on page 633
- "NLPCTIw.d Informat" on page 635

NLPCTIw.d Format

Writes percentage data of the international expression in the specified locale.

**Categories:**
- CAS
- Numeric

**Alignment:**
- Left

**Syntax**

NLPCTIw.d

**Syntax Description**

**w**

specifies the width of the output field.

- Default: 6
- Range: 4–32

**d**

specifies to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.

- Default: 0
- Range: 0–31
Comparisons

The NLPCTIw.d format writes percentage data of the international expression in the specified locale. The NLPCTw.d format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTIw.d format writes the percentage value with a comma (,) as thousands separator and a period (.) as a decimal separator.

The NLPCTw.d format is similar to the PERCENTw.d format except the NLPCTw.d format is locale-specific.

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-12.3456789,nlpcti32.2);
y=put(-12.3456789,percent32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>( 1234.57)</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLPCTw.d Format” on page 283

Informat:
- “NLPCTIw.d Informat” on page 635

NLPCTNw.d Format

Produces percentages, using a minus sign for negative values.

<table>
<thead>
<tr>
<th>Categories</th>
<th>CAS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Numeric</td>
</tr>
</tbody>
</table>

| Alignment    | Right     |

Syntax

NLPCTNw.d
Syntax Description

\texttt{w}

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>4–32</td>
</tr>
</tbody>
</table>

Tip

The width of the output field must account for the minus sign ( – ), the percent sign ( % ), and a trailing blank, whether the number is negative or positive.

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value. This argument is optional.

<table>
<thead>
<tr>
<th>Default</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>0–31</td>
</tr>
<tr>
<td>Requirement</td>
<td>must be less than \texttt{w}</td>
</tr>
</tbody>
</table>

Details

The NLPCTN\texttt{w.d} format multiplies negative values by 100, adds a minus sign to the beginning of the value, and adds a percent sign (%) to the end of the formatted value.

Example

\[x=-0.02;
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x nlpctn6.;</td>
<td>(x=-2%)</td>
</tr>
<tr>
<td>put x percentn6.;</td>
<td>(x=-2%)</td>
</tr>
</tbody>
</table>

NLPCTPw.d Format

Writes locale-specific numeric values as percentages.

Categories: CAS, Numeric

Alignment: Right
Syntax

\textbf{NLPCTP}_{w.d}

Syntax Description

\textbf{w}

specifies the width of the output field.

Default: 6

Range: 4–32

Tip: The width of the output field must account for the percent sign (%).

\textbf{d}

specifies the number of digits to the right of the decimal point in the numeric value. This argument is optional. The thousands separator and decimal symbol for the NLPCTP format is locale-specific.

Default: 0

Range: 0–31

Requirement: must be less than \textbf{w}

Details

The NLPCTP_{w.d} format multiplies values by 100, formats them, and adds a percent sign (%) to the end of the formatted value. The NLPCTP_{w.d} format is similar to the PERCENT_{w.d} format, except that the thousands separator and decimal symbol for the NLPCTP_{w.d} format is locale-specific.

Example

\begin{verbatim}
x = 0.02;
\end{verbatim}

\begin{tabular}{ll}
\textbf{Statements} & \textbf{Results} \\
put x nlpctp6.; & \(-2\%\) \\
put x percent6.; & \((2\%)\) \\
\end{tabular}

\textbf{NLPVALUE}_{w.d} Format

Writes p-values of the local expression in the specified locale.

Categories: CAS, Numeric

Alignment: Left
Syntax

**NLPVALUEw.d**

Syntax Description

**w**

specifies the width of the output field.

Default: 6

Range: 3–32

**d**

specifies to divide the number by $10^d$. If the data contains decimal separators, the $d$ value is ignored.

Default: 4

Range: 1–30

Example

This example uses the german_Germany locale option.

Statements:

```sas
options locale=german_germany;
data _null_
  put "+--- nlpvalue min=3 default=6 max=32 ---+";
  x=0.1248;
  put x= +5 x pvalue. +5 x nlpvalue.;
  put x= +5 x pvalue3.1 +5 x nlpvalue3.1;
  put x= +5 x pvalue20.2 +5 x nlpvalue20.2;
  put x= +5 x pvalue32.3 +5 x nlpvalue32.3;
run;
```

Results:

```
+--- nlpvalue min=3 default=6 max=32 ---+
x=0.1248   0.1248     0,1248
x=0.1248   0.1     0,1
x=0.1248                   0.12                     0,12
x=0.1248                              0.125                           0,125
```

See Also

Format:

- “PVALUEw.d Format” in *SAS Formats and Informats: Reference*
NLSTRMONw.d Format

Writes the month name in the specified locale.

Categories: CAS
            Numeric
Alignment: Left

Syntax

NLSTRMONw.d

Syntax Description

w
  specifies the width of the output field
  Default  20
  Range    1-200

d
  specifies the following:
    00000001: write abbreviated form.
    00000010: write capitalized form.
  Default  0
  Range    0-3

Details

The NLSTRMONw.d format writes a SAS value, 1–12 as the name-of-the-month in the specified locale. The following examples use the English_UnitedStates locale.

  1 = the first month (January)
  2 = the second month (February)
  3 = the third month (March)
  4 = the fourth month (April)
  5 = the fifth month (May)
  6 = the sixth month (June)
  7 = the seventh month (July)
  8 = the eight month (August)
  9 = the ninth month (September)
 10 = the 10th month (October)
11 = the 11th month (November)
12 = the 12th month (December)

Example
This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>monnum = 1; /* January=1, December=12 */</td>
<td></td>
</tr>
<tr>
<td>put monnum NLSTRMON20. ;</td>
<td>January</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.1; /* decimal .1 specified use abbreviation */</td>
<td>Jan</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.2;</td>
<td>JANUARY</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.3;</td>
<td>JAN</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLSTRQTRw.d Format

Writes a numeric value as the quarter-of-the-year in the specified locale.

Categories: CAS
Numeric
Alignment: Left

Syntax

`NLSTRQTRw.d`

Syntax Description

w
specifies the width of the output field

Default 20
Range 1–200

d
specifies the following:
- 00000001: write abbreviated form.
- 00000010: write capitalized form.
Details
The NLSTRQTRw.d format writes a SAS value, 1–4 as the name-of-the-quarter for the year in the specified locale. The following examples use the English_UnitedStates locale.

- 1 = 1st quarter
- 2 = 2nd quarter
- 3 = 3rd quarter
- 4 = 4th quarter

Example
This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>qtrnum = 1; /* January=1, December=12 */</td>
<td>1st quarter</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20. ;</td>
<td>Q1</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.1; /* decimal .1 specified use abbreviation. */</td>
<td>1ST QUARTER</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.2;</td>
<td>Q1</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.3;</td>
<td></td>
</tr>
</tbody>
</table>

### NLSTRWKw.d Format
Writes a numeric value as the day-of-the-week in the specified locale.

**Categories:**
- CAS
- Numeric

**Alignment:**
- Left

**Syntax**

```
NLSTRWKw.d
```
Syntax Description

**w**

specifies the width of the output field

- Default: 20
- Range: 1–200

**d**

specifies the following:

- 00000001: write abbreviated form.
- 00000010: write capitalized form.

- Default: 0
- Range: 0–3

Details

The NLSTRWKw.d format writes a SAS value, 1–7 as the name-of-the-week in the specified locale. The following examples use the English_UnitedStates locale.

- **1** = First day-of-week (Monday)
- **2** = Second day-of-week (Tuesday)
- **3** = Third day-of-week (Wednesday)
- **4** = Fourth day-of-week (Thursday)
- **5** = Fifth day-of-week (Friday)
- **6** = Sixth day-of-week (Saturday)
- **7** = Seventh day-of-week (Sunday)

Example

This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>; wknum = 1 ; /* Sunday=1, Saturday=7 <em>/ put wknum NLSTRWK20. ; put wknum NLSTRWK20.1; /</em> decimal .1 specified use abbreviation. */ put wknum NLSTRWK20.2; put wknum NLSTRWK20.3; run;</td>
<td>Sunday Sun SUNDAY SUN</td>
</tr>
</tbody>
</table>
NLTIMAPw. Format

Converts a SAS time value to the time value of a specified locale, and then writes the value as a time value with a.m. or p.m. NLTIMAP also converts SAS date-time values.

<table>
<thead>
<tr>
<th>Categories:</th>
<th>CAS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Date and Time</td>
</tr>
<tr>
<td></td>
<td>Time</td>
</tr>
</tbody>
</table>

| Alignment:      | Left     |

Syntax

NLTIMAPw.

Syntax Description

w

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>4–200</td>
</tr>
</tbody>
</table>

Details

The NLTIMAP format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 85 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Comparisons

The NLTIMAPw. format is similar to the TIMEAMPMw. format, except that the NLTIMAPw. format is locale-specific.

Example

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>4:24:43 PM</td>
</tr>
<tr>
<td>put time nltimap.;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Statements | Results
options locale=German_Germany;  | 4:24:43 nachm
put time nltimap14.;

See Also

Format:
- "NLTIMEw. Format" on page 295

NLTIMEw. Format

Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value. NLTIME also converts SAS date-time values.

Categories:
- CAS
- Date and Time
- Time

Alignment:
- Left

Syntax

NLTIMEw.

Syntax Description

w
 specifies the width of the input field.

Default 20
Range 10–200

Comparisons

The NLTIMEw. format is similar to the TIMEw. format, except that the NLTIMEw. format is locale-specific.

Example

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.
Statements | Results
--- | ---

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>4:24:43</td>
</tr>
<tr>
<td>put time nlt ime.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>16.24</td>
</tr>
<tr>
<td>put time nlt ime.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

**Format:**
- “NLTIMAPw. Format” on page 294

**NLTIMELw. Format**

Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value in a long-uniform pattern.

**Categories:**
- CAS
- Date and Time

**Alignment:**
- Left

**Restriction:**
- This function runs in SAS Viya 3.5 and is not supported in SAS 9.

**Syntax**

**NLTIMELw.**

**Optional Argument**

<table>
<thead>
<tr>
<th>w</th>
<th>specifies the width of the input field. If necessary, SAS abbreviates the time to fit the format width.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>30</td>
</tr>
<tr>
<td>Range</td>
<td>10–200</td>
</tr>
</tbody>
</table>

**Details**

The NLTIMELw. format writes the time in a long-uniform pattern.
Examples:

Example 1

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

data a;
  options locale=English_UnitedStates;
  options timezone='America/Chicago';
  time=59083;
  put time nltimel.;
  put time nltimel10.;
  put time nltimel200.;
run;

04:24:43 PM CDT
04:24 PM
04:24:43 PM CDT

Example 2

data b;
  options locale=German_Germany;
  options timezone='Europe/Berlin';
  time=59083;
  put time nltimel.;
  put time nltimel10.;
  put time nltimel12.;
  put time nltimel18.;
  put time nltimel200.;
run;

16:24:43 CEST
16:24:43
16:24:43
16:24:43 CEST
16:24:43 CEST

Example 3

data c;
  options locale=English_UnitedStates;
  options timezone='Etc/GMT-6';
  time=59083;
  put time nltimel.;
  put time nltimel10.;
  put time nltimel200.;
run;

04:24:43 PM GMT+6
04:24 PM
04:24:43 PM GMT+6
NLTIMEMw. Format

Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value in a medium-uniform pattern.

Categories:  
- CAS
- Date and Time

Alignment:  
Left

Restriction:  
This function runs in SAS Viya 3.5 and is not supported in SAS 9.

Syntax

NLTIMEMw.

Optional Argument

w  
specifies the width of the input field. If necessary, SAS abbreviates the time to fit the format width.

Default  
14

Range  
10–200

Details

The NLTIMEMw. format writes the time in a medium-uniform pattern.

Example

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

data a;
  options locale=English_UnitedStates;
    time=59083;
    put time nltimem.;
    put time nltimem10.;
    put time nltimem200.;
  run;

04:24:43 PM
04:24 PM
04:24:43 PM

data b;
  options locale=German_Germany;
    time=59083;
    put time nltimem.;
    put time nltimem10.;
NLTIMESw. Format

Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value in a short-uniform pattern.

Categories: CAS
            Date and Time

Alignment: Left

Restriction: This function runs in SAS Viya 3.5 and is not supported in SAS 9.

Syntax

NLTIMESw.

Optional Argument

w.

specifies the width of the input field. If necessary, SAS abbreviates the time to fit the format width.

Default 11
Range 10–200

Details

The NLTIMESw. format writes the time in a short-uniform pattern.

Example

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

data a;
  options locale=English_UnitedStates;
  time=59083;
  put time nltimes.;
  put time nltimes10.;
  put time nltimes200.;
data b;
  options locale=German_Germany;
  time=59083;
  put time nltimes.;
  put time nltimes10.;
  put time nltimes12.;
  put time nltimes18.;
  put time nltimes200.;
run;

$UCS2Bw. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 16-bit, UCS2, Unicode encoding.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$UCS2B

Syntax Description

w
  specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

  Default  8
  Range    2–32767

Details

The $UCS2Bw. format writes a character string in big-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.
Comparisons

The $UCS2Bw. format performs processing that is the opposite of the $UCS2BEw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>5927</td>
</tr>
<tr>
<td>x = 'Ａ';</td>
<td></td>
</tr>
<tr>
<td>y=put(x,$ucs2b2.);</td>
<td></td>
</tr>
<tr>
<td>put y $hex.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

**Formats:**
- "$UCS2Lw. Format" on page 303
- "$UCS2Xw. Format" on page 305
- "$UTF8Xw. Format" on page 323
- "$UCS2BEw. Format" on page 301

**Informats:**
- "$UCS2Bw. Informat" on page 641
- "$UCS2BEw. Informat" on page 642
- "$UCS2Lw. Informat" on page 643
- "$UCS2Xw. Informat" on page 646
- "$UTF8Xw. Informat" on page 661

$UCS2BEw. Format

Processes a character string that is in big-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character

Alignment: Left

Restriction: This format is not supported in a DATA step that runs in CAS.
Syntax

$UCS2BE_{w}.$

Syntax Description

\( w \)

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8

Range 2–32767

Details

The $UCS2BE_{w}. format writes a character string in the encoding of the current SAS session. It processes character strings that are in big-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

Comparisons

The $UCS2BE_{w}. format performs processing that is the opposite of the $UCS2B_{w}. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '592700410042'x; put x $ucs2be.;</td>
<td>𐀁AB</td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$UCS2B{w}. Format" on page 300

Informats:

- "$UCS2B{w}. Informat" on page 641
- "$UCS2BE{w}. Informat" on page 642
$UCS2Lw. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 16-bit, UCS2, Unicode encoding.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$UCS2Lw.

Syntax Description

w
specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8
Range 2–32767

Details

The $UCS2Lw. format writes a character string in little-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

Comparisons

The $UCS2Lw. format performs processing that is the opposite of the $UCS2LEw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1</td>
<td>--------</td>
</tr>
</tbody>
</table>


Statements | Result
--- | ---
data _null_; | 
\(x = '大';\) | 
y=put(x,$ucs2l2.); | 
put y $hex.; | 
run; | 
2759 | 

See Also

**Formats:**
- "$UCS2Bw. Format" on page 300
- "$UCS2LEw. Format" on page 304
- "$UCS2Xw. Format" on page 305
- "$UTF8Xw. Format" on page 323

**Informats:**
- "$UCS2Bw. Informat" on page 641
- "$UCS2Lw. Informat" on page 643
- "$UCS2LEw. Informat" on page 645
- "$UCS2Xw. Informat" on page 646
- "$UTF8Xw. Informat" on page 661

**$UCS2LEw. Format**

Processes a character string that is in little-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

**Category:** Character

**Alignment:** Left

**Restriction:** This format is not supported in a DATA step that runs in CAS.

**Syntax**

\$UCS2LEw.

**Syntax Description**

\(w\)

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.
Default 8
Range 2–32767

Details
The $UCS2LEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in little-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

Comparisons
The $UCS2LEw. format performs processing that is the opposite of the $UCS2Lw. format.

Example
This example uses the Japanese Shift-JIS encoding, which is supported under the UNIX operating environment.

```
x = '275941004200'x;
put x $ucs2le.;
```

See Also

Format:
- “$UCS2Lw. Format” on page 303

Informat:
- “$UCS2Lw. Informat” on page 643
- “$UCS2LEw. Informat” on page 645

$UCS2Xw. Format
Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 16-bit, UCS2, Unicode encoding.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.
Syntax

$UCS2Xw.

Syntax Description

w
specifies the width of the output field. Specify enough width to accommodate the
16-bit size of the Unicode characters.

Default 8
Range 2–32767

Details

The $UCS2Xw. format writes a character string in 16-bit, UCS2 (universal character
set code in two octets), Unicode encoding, by using byte order that is native to the
operating environment.

Comparisons

The $UCS2Xw. format performs processing that is the opposite of the $UCS2XEw.
format. If you are exchanging data within the same operating environment, use the
$UCS2Xw. format. If you are exchanging data with a different operating
environment, use the $UCS2Bw. format or $UCS2Lw. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported
under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '夫';</td>
<td>'5927'x (binary) or '2759'x (little endian)</td>
</tr>
<tr>
<td>put x $ucs2x2.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$UCS2Bw. Format" on page 300
- "$UCS2XEw. Format" on page 307
- "$UCS2Lw. Format" on page 303
- "$UTF8Xw. Format" on page 323

Informats:

- "$UCS2Bw. Informat" on page 641
$UCS2XEw. Format

Processes a character string that is in native-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

\$UCS2XEw.

Syntax Description

\textit{w}

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default \hspace{1cm} 8

Range \hspace{1cm} 2–32767

Details

The $UCS2XEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in native-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

Comparisons

The $UCS2XEw. format performs processing that is the opposite of the $UCS2Xw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>\hspace{2cm}</td>
<td>\hspace{2cm}</td>
</tr>
</tbody>
</table>

\hspace{2cm}
Statements | Result
--- | ---
x = 'e5a4a7'x; /* Japanese '/tcp in UTF8 */;
put x $utf8xe10.;

See Also

Format:
- "$UCS2Xw. Format" on page 305

Informats:
- "$UCS2Xw. Informat" on page 646
- "$UCS2XEw. Informat" on page 647

$UCS4Bw. Format
Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 32-bit, UCS4, Unicode encoding.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax
$UCS4Bw.

Syntax Description

w
specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8
Range 4–32767

Details
The $UCS4Bw. format writes a character string in big-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.
Comparisons

The $UCS4Bw. format performs processing that is the opposite of the $UCS4BEw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '十';</td>
<td>'00005927'x (binary)</td>
</tr>
<tr>
<td>put x $ucs4b4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UCS2Lw. Format" on page 303
- "$UCS2Xw. Format" on page 305
- "$UCS4BEw. Format" on page 309
- "$UCS4Lw. Format" on page 311
- "$UCS4Xw. Format" on page 313
- "$UTF8Xw. Format" on page 323

Informat:
- "$UCS2Bw. Informat" on page 641
- "$UCS2Lw. Informat" on page 643
- "$UCS2Xw. Informat" on page 646
- "$UCS4Bw. Informat" on page 648
- "$UCS4Lw. Informat" on page 649
- "$UCS4Xw. Informat" on page 650
- "$UTF8Xw. Informat" on page 661

$UCS4BEw. Format

Processes a character string that is in big-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

\$UCS4BEw.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8

Range 4–32767

Details

The \$UCS4BEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in big-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

Comparisons

The \$UCS4BEw. format performs processing that is the opposite of the \$UCS4Bw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '000059270000004100000042'x; put x $ucs4be.;</td>
<td>ฆ�</td>
</tr>
</tbody>
</table>

See Also

Format:
- "$UCS4Bw. Format" on page 308

Informat:
- "$UCS4Bw. Informat" on page 648
**$UCS4Lw. Format**

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 32-bit, UCS4, Unicode encoding.

**Category:** Character  
**Alignment:** Left  
**Restriction:** This format is not supported in a DATA step that runs in CAS.

**Syntax**

$UCS4Lw.$

**Syntax Description**

$w$

Specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

**Default** 8  
**Range** 4–32767

**Details**

The $UCS4Lw.$ format writes a character string in little-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

**Comparisons**

The $UCS4Lw.$ format performs processing that is the opposite of the $UCS4LEw.$ format.

**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1</td>
<td></td>
</tr>
</tbody>
</table>


Statements | Result
--- | ---
data _null_; 2759
x = 'ä';
y=put(x, $ucs4l4.);
put y $hex.;
run;

See Also

Formats:

- "$UCS2Bw. Format" on page 300
- "$UCS2Xw. Format" on page 305
- "$UCS4Bw. Format" on page 308
- "$UCS4LEw. Format" on page 312
- "$UCS4Xw. Format" on page 313
- "$UTF8Xw. Format" on page 323

Informats:

- "$UCS2Bw. Informat" on page 641
- "$UCS2Lw. Informat" on page 643
- "$UCS2Xw. Informat" on page 646
- "$UCS4Bw. Informat" on page 648
- "$UCS4Lw. Informat" on page 649
- "$UCS4Xw. Informat" on page 650
- "$UTF8Xw. Informat" on page 661

$UCS4LEw. Format

Processes a character string that is in little-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$UCS4LEw.
Syntax Description

$ w$
specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8
Range 4–32767

Details

The $UCS4LEw$. format writes a character string in the encoding of the current SAS session. It processes character strings that are in little-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

Comparisons

The $UCS4LEw$. format performs processing that is the opposite of the $UCS4Lw$. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '27590000410000042000000'x;</td>
<td>AB</td>
</tr>
<tr>
<td>put x $ucs4le.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- “$UCS4Lw. Format” on page 311

Informat:
- “$UCS4Lw. Informat” on page 649

$UCS4Xw. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 32-bit, UCS4, Unicode encoding.

Category: Character
Syntax

$UCS4Xw.

Syntax Description

\( w \)

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8  
Range 4–32767

Details

The $UCS4Xw. format writes a character string in 32-bit, UCS4 (universal character set code in two octets), Unicode encoding, by using byte order that is native to the operating environment.

Comparisons

The $UCS4Xw. format performs processing that is the opposite of the $UCS4XEw. format. If you are exchanging data within the same operating environment, use the $UCS4Xw. format. If you are exchanging data with a different operating environment, use the $UCS4Bw. format or $UCS4Lw. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 'Ａ';</td>
<td>'00005927'x (binary) or '27590000'x (little endian)</td>
</tr>
<tr>
<td>put x $ucs4x4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$UCS2Lw. Format" on page 303
- "$UCS4XEw. Format" on page 315
- "$UCS2Xw. Format" on page 305
$UCS4XEw. Format

Processes a character string that is in native-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$UCS4XEw.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8
Range 4–32767

Details

The $UCS4XEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in native-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.
Comparisons
The $UCS4XEw. format performs processing that is the opposite of the $UCS4Xw. format.

Example
This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = '275900004100000042000000'x;</code></td>
<td>*AB (little endian)</td>
</tr>
<tr>
<td><code>put x $ucs4be4.;</code></td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- "$UCS4Xw. Format" on page 313

Informat:
- "$UCS4Xw. Informat" on page 650

$UESCw. Format
Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode escape (UESC) representation.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax
$UESCW.

Syntax Description

$UESCW.

w specifies the width of the input field.

Default 8
Details
If the characters are not available on all operating environments, for example, 0–9, a–z, A–Z, they must be represented in UESC. $UESCw. can be nested.

Comparisons
The $UESCw. format performs processing that is opposite of the $UESCEw. format.

Example
This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='夏' ;</td>
<td>¥u5927</td>
</tr>
<tr>
<td>y='u5927'</td>
<td>¥u5927</td>
</tr>
<tr>
<td>z='uu5927' ;</td>
<td>¥uu5927</td>
</tr>
<tr>
<td>put x = $uesc10. ;</td>
<td></td>
</tr>
<tr>
<td>put y = $uesc10. ;</td>
<td></td>
</tr>
<tr>
<td>put z = $uesc10. ;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UESCEw. Format" on page 317

Informats:
- "$UESCw. Informat" on page 652
- "$UESCEw. Informat" on page 654

$UESCEw. Format

Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.
Syntax
$UESCEw.

Syntax Description

w
specifies the width of the output field.

Default 8

Range 1–32767

Details
If the data is not supported by the encoding of the current SAS session, the data remains in UESC.

Comparisons
The $UESCEw. format performs processing that is the opposite of the $UESCw. format.

Example
This example uses the Japanese Shift-JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put('¥u5927',$uesce10.) ;</td>
<td>x=¥u5927</td>
</tr>
<tr>
<td>x=put('¥uu5927',$uesce10.) ;</td>
<td>x=¥u5927</td>
</tr>
<tr>
<td>x=put('¥uuu5927',$uesce10.) ;</td>
<td>x=¥u5927</td>
</tr>
</tbody>
</table>

See Also

Format:
- "$UESCw. Format" on page 316

Informats:
- "$UESCw. Informat" on page 652
- "$UESCEw. Informat" on page 654
$UNCRw. Format

Processes a character string that is encoded in the current SAS session, and then writes the character string in numeric character representation (NCR).

**Category:** Character

**Alignment:** Left

**Restriction:** This format is not supported in a DATA step that runs in CAS.

### Syntax

$UNCRw.

### Syntax Description

**w**

specifies the width of the output field.

**Default:** 8

**Range:** 1–32767

### Comparisons

The $UNCRw. format performs processing that is the opposite of the $UNCREw. format.

### Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='91E5'x ; /* Japanese 'ァ' in Shift-JIS */</td>
<td>ぁ</td>
</tr>
<tr>
<td>y='abc';</td>
<td>abc</td>
</tr>
<tr>
<td>put x $uncr10.;</td>
<td></td>
</tr>
<tr>
<td>put y $uncr10.;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Formats:**
$UNCREw. Format

Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$UNCREw.

Syntax Description

w

specifies the width of the output field.

Default 8
Range 1–32767

Details

National characters should be represented in NCR.

Comparisons

The $UNCREw. format performs processing that is the opposite of the $UNCRw. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------</td>
</tr>
</tbody>
</table>


$\text{UPARENw}. \text{ Format} \quad \text{321}

$\text{UPARENw}. \text{ Format}

Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
<tr>
<td>Restriction:</td>
<td>This format is not supported in a DATA step that runs in CAS.</td>
</tr>
</tbody>
</table>

Syntax

$\text{UPARENw}$.w

Syntax Description

w

specifies the width of the output field.

Default 8

Range 7–32767

Details

The character string is encoded with parentheses and Unicode hexadecimal representation.

Comparisons

The $\text{UPARENw}. \text{ format}$ performs processing that is the opposite of the $\text{UPARENEw}. \text{ format}$. 

See Also

Formats:

- "$\text{UNCRw}. \text{ Format}$" on page 319

Informats:

- "$\text{UNCRw}. \text{ Informat}$" on page 655
- "$\text{UNCREw}. \text{ Informat}$" on page 656

Examples

\begin{verbatim}
x='&22823;abc';
put x $uncr10.;
\end{verbatim}

Results

abc
Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{x=' '}; \texttt{y='abc3'}; \texttt{put x $uparen7.;}; \texttt{put y $uparen28.;}</td>
<td>\texttt{&lt;u5927&gt;} &lt;u0061&gt; &lt;u0062&gt; &lt;u0063&gt; &lt;u0033&gt;</td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UPARENEw. Format" on page 322

Informats:
- "$UPARENw. Informat" on page 657
- "$UPARENEw. Informat" on page 658

$UPARENEw. Format

Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

\texttt{$UPARENEw}$

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 8
Range 1–32767
Comparisons

The $UPARENEw. format performs processing that is the opposite of the $UPARENw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='&lt;u0061&gt;&lt;u0062&gt;&lt;u0063&gt;&lt;u0033&gt;'; put x $uparene4.;</td>
<td>abc3</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UPARENw. Format” on page 321

Informs:
- “$UPARENw. Informat” on page 657
- “$UPARENEw. Informat” on page 658

$UTF8Xw. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in universal transformation format (UTF-8) encoding.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax

$UTF8Xw.

Syntax Description

w

specifies the width of the output field. Specify enough width to include all of the characters in the variable. The width of the characters is dependent on the code point value of the individual characters.
Default: 8
Range: 1–32767

Comparisons
This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '91E5'x; /* Japanese '¥' in Shift-JIS */ put x $utf8x10.;</td>
<td>x='e5a4a7'x</td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UCS2Bw. Format" on page 300
- "$UCS2Lw. Format" on page 303
- "$UCS2Xw. Format" on page 305

Informat:
- "$UCS2Bw. Informat" on page 641
- "$UCS2Lw. Informat" on page 643
- "$UCS2Xw. Informat" on page 646

$UTF8XEw. Format
Processes a character string that is in universal transformation format (UTF-8), and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left
Restriction: This format is not supported in a DATA step that runs in CAS.

Syntax
$UTF8XEw.
Syntax Description

\( w \)

specifies the width of the output field. Specify enough width to include all of the characters in the variable. The width of the characters is dependent on the code point value of the individual characters.

- **Default**: 8
- **Range**: 1–32767

Comparisons

The $UTF8XEw. format performs processing that is the opposite to the $UTF8Xw. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = unicode('u5927'); put x $utf8xe10.;</td>
<td>丸</td>
</tr>
</tbody>
</table>

See Also

**Formats:**

- "$UTF8Xw. Format" on page 323

**Informats:**

- "$UTF8Xw. Informat" on page 661

$VSLOGw. Format

Processes a character string that is in visual order, and then writes the character string in left-to-right logical order.

- **Category**: BIDI Text Handling
- **Alignment**: Left
- **Restriction**: This format is not supported in a DATA step that runs in CAS.
Syntax

$VSLOGw.

Syntax Description

$w

specifies the width of the output field.

Default 200

Range 1–32767

Details

The $VSLOGw. format is used when transferring data that is stored in visual order. An example is transferring data from a UNIX server to a Windows client.

Note: The $VSLOGw. format does not correctly process all combinations of data strings.

Comparisons

The $VSLOGw. format performs processing that is opposite to the $VSLOGRw. format.

Example

The following example uses the Hebrew input value of "-flight".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $vslog12.;</td>
<td>נסיע.club</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "computer".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $vslog12.;</td>
<td>ذاب.computer</td>
</tr>
</tbody>
</table>

See Also

Format:
$VSLOGRw. Format

Processes a character string that is in visual order, and then writes the character string in right-to-left logical order.

- **Category:** BIDI Text Handling
- **Alignment:** Left
- **Restriction:** This format is not supported in a DATA step that runs in CAS.

### Syntax

**$VSLOGRw.**

### Syntax Description

- **$w** specifies the width of the output field.
  - **Default:** 200
  - **Range:** 1–32767

### Details

The $VSLOGRw. format is used when transferring data that is stored in visual order. An example is transferring data from a UNIX server to a Windows client.

**Note:** The $VSLOGRw. format does not correctly process all combinations of data strings.

### Comparisons

The $VSLOGRw. format performs processing that is opposite to the $VSLOGw. format.

### Example

The following example uses the Hebrew input value of "_flight."
The following example uses the Arabic input value of "computer."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12;</td>
<td>flight تاينر</td>
</tr>
</tbody>
</table>

See Also

**Informat**:  
- "$VSLOGw. Informat" on page 662  
- "$VSLOGRw. Informat" on page 663

---

**WEEKUw. Format**

Writes a week number in decimal format by using the U algorithm.

**Categories**: CAS  
Date and Time

**Alignment**: Left

**Syntax**

```
WEEKUw.
```

**Syntax Description**

`w`

specifies the width of the output field.

- **Default**: 11
- **Range**: 2–200
Details

The WEEKUw. format writes a week-number format. The WEEKUw. format writes the various formats depending on the specified width. Algorithm U calculates the SAS date value by using the number of the week within the year (Sunday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yyWwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWwwdd</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>

Comparisons

The WEEKVw. format writes the week number as a decimal number in the range 01–53, with weeks that begin on a Monday and week 1 of the year including both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKWw. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The WEEKUw. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Example

sasdate = '01JAN2003'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,weeku3.);</td>
<td>W00</td>
</tr>
<tr>
<td>w=put(sasdate,weeku5.);</td>
<td>03W00</td>
</tr>
<tr>
<td>x=put(sasdate,weeku7.);</td>
<td>03W0004</td>
</tr>
<tr>
<td>y=put(sasdate,weeku9.);</td>
<td>2003W0004</td>
</tr>
<tr>
<td>z=put(sasdate,weeku11.);</td>
<td>2003-W00-04</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:
- “WEEKVw. Format” on page 330
- “WEEKWw. Format” on page 331

WEEKVw. Format

Writes a week number in decimal format by using the V algorithm.

Categories:
- CAS
- Date and Time

Alignment:
- Left

Syntax

\texttt{WEEKVw.}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default: 11

Range: 2–200

Details

The WEEKVw. format writes the various formats depending on the specified width. Algorithm V calculates the SAS date value, and the number-of-the-week value is represented as a decimal number in the range 01–53, with a leading zero and maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For example, the fifth week of the year would be represented as 06.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>03W01</td>
</tr>
</tbody>
</table>
Comparisons

The WEEKVw. format writes the week number as a decimal number in the range 01–53, with weeks that begin on a Monday and week 1 of the year including both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKWw. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The WEEKUw. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Example

```sas
sasdate='01JAN2003'd;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,weekv3.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate,weekv5.);</td>
<td>03W01</td>
</tr>
<tr>
<td>x=put(sasdate,weekv7.);</td>
<td>03W0103</td>
</tr>
<tr>
<td>y=put(sasdate,weekv9.);</td>
<td>2003W0103</td>
</tr>
<tr>
<td>z=put(sasdate,weekv11.);</td>
<td>2003-W01-03</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

- “WEEKUw. Format” on page 328
- “WEEKWw. Format” on page 331

WEEKWw. Format

Writes a week number in decimal format by using the W algorithm.
Syntax

\textbf{WEEKW}_w.

Syntax Description

\textit{w}

specifies the width of the output field.

Default 11

Range 2–200

Details

The \textit{WEEKW}_w. format writes the various formats depending on the specified width. Algorithm W calculates the SAS date value using the number of the week within the year (Monday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>\textit{Www}</td>
<td>\textit{w01}</td>
</tr>
<tr>
<td>5-6</td>
<td>\textit{yyWww}</td>
<td>\textit{03W01}</td>
</tr>
<tr>
<td>7-8</td>
<td>\textit{yyWwwdd}</td>
<td>\textit{03W0101}</td>
</tr>
<tr>
<td>9-10</td>
<td>\textit{yyyyWwwdd}</td>
<td>\textit{2003W0101}</td>
</tr>
<tr>
<td>11-200</td>
<td>\textit{yyyy-Www-dd}</td>
<td>\textit{2003-W01-01}</td>
</tr>
</tbody>
</table>

Comparisons

The \textit{WEEKV}_w. format writes the week number as a decimal number in the range 01–53. Weeks beginning on a Monday and on week 1 of the year include both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The \textit{WEEKW}_w. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The \textit{WEEKU}_w. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.
Example

sasdate = '01JAN2003'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,weekw3.);</td>
<td>W03</td>
</tr>
<tr>
<td>w=put(sasdate,weekw5.);</td>
<td>03W03</td>
</tr>
<tr>
<td>x=put(sasdate,weekw7.);</td>
<td>03W0003</td>
</tr>
<tr>
<td>y=put(sasdate,weekw9.);</td>
<td>2003W0003</td>
</tr>
<tr>
<td>z=put(sasdate,weekw11.);</td>
<td>2003-W00-03</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "WEEKUw. Format" on page 328
- "WEEKVw. Format" on page 330

YENw.d Format

Writes numeric values with yen signs, commas, and decimal points.

Category: Numeric

Alignment: Right

Restrictions: This format is not supported in a DATA step that runs in CAS. The YEN format does not support the UTF-8 encoding. See the NLMNJPY format for this functionality.

Syntax

YENw.d

Syntax Description

w

specifies the width of the output field.

Default  8

Range  1–32
specifies the number of digits to the right of the decimal point in the numeric value.

Range  0–31

Details

The YENw.d format writes numeric values with a leading yen sign and with a comma that separates every three digits of each value.

The hexadecimal representation of the code for the yen sign character is 5B on EBCDIC systems and 5C on ASCII systems. The monetary character these codes represent might be different in other countries.

Example

```plaintext
put cost yen10.2;
data _null_;    value=1254.71;
    put value yen10.2;
run;
```

<table>
<thead>
<tr>
<th>Cost</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>1254.71</td>
<td>¥1,254.71</td>
</tr>
</tbody>
</table>

See Also

- **Informat:**
  - "YENw.d Informat" on page 665

### YYWEEKUw. Format

Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.

- **Categories:** CAS
  - Date and Time

- **Alignment:** Left

**Syntax**

YYWEEKUw.
Syntax Description

w
  specifies the width of the output field.

  Default  7
  Range    2-8

Details

The YYWEEKUw. format writes a week-number format. The YYWEEKUw. format writes the various formats depending on the specified width. Algorithm U calculates the SAS date value by using the number of the week within the year (Sunday is considered the first day of the week).

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>W01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWww</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-Www</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Comparisons

The YYWEEKUw. format is similar to the WEEKUw. format except that the YYWEEKUw. format does not specify the day-of-week information. Also, the YYWEEKUw. format does not accept any width that is greater than 8.

Example

```sas
date = '01JAN2007'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| ----+----1----+```
YYWEEKVw. Format

Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.

**Syntax**

**YYWEEKVw.**

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 7
  - Range: 2–8

**Details**

The YYWEEKVw. format writes the various formats depending on the specified width. Algorithm V calculates the SAS date value, and the number-of-the-week value is represented as a decimal number in the range 01–53, with a leading 0 and maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first
Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For example, the fifth week of the year would be represented as 06.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWww</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-Www</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Comparisons

The YYWEEKVw. format is similar to the WEEKVw. format except that the YYWEEKVw. format does not specify the day-of-week information. Also, the YYWEEKVw. format does not accept a width that is greater than 8.

Example

sasdate = '01JAN2007'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweekv3.);</td>
<td>W01</td>
</tr>
<tr>
<td>v=put(sasdate,yyweekv4.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate,yyweekv5.);</td>
<td>07W01</td>
</tr>
<tr>
<td>x=put(sasdate,yyweekv6.);</td>
<td>07W01</td>
</tr>
<tr>
<td>y=put(sasdate,yyweekv7.);</td>
<td>2007W01</td>
</tr>
<tr>
<td>z=put(sasdate,yyweekv8.);</td>
<td>2007-W01</td>
</tr>
<tr>
<td>put u;</td>
<td></td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- "WEEKVw. Format" on page 330
YYWEEKWw. Format

Writes a week number in decimal format by using the W algorithm, excluding the day-of-week information.

Categories: CAS
         Date and Time

Alignment: Left

Syntax

YYWEEKWw.

Syntax Description

w
specifies the width of the output field.

Default  7

Range    2–8

Details

The YYWEEKWw. format writes the various formats depending on the specified width. Algorithm W calculates the SAS date value using the number of the week within the year.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>WWW</td>
<td>W01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyyyWWW</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWWW</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-WWW</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Comparisons

The YYWEEKWw. format is similar to the WEEKWw. format except that the YYWEEKWw. format does not specify the day-of-week information. Also, the YYWEEKWw. format does not accept any width that is greater than 8.
Example

sasdate = '01JAN2007'd

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweek3.);</td>
<td>W01</td>
</tr>
<tr>
<td>v=put(sasdate,yyweek4.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate,yyweek5.);</td>
<td>07W01</td>
</tr>
<tr>
<td>x=put(sasdate,yyweek6.);</td>
<td>07W01</td>
</tr>
<tr>
<td>y=put(sasdate,yyweek7.);</td>
<td>2007W01</td>
</tr>
<tr>
<td>z=put(sasdate,yyweek8.);</td>
<td>2007-W01</td>
</tr>
<tr>
<td>put u;</td>
<td></td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- "WEEKWw. Format" on page 331
PART 5

Functions for NLS

Chapter 11
Internationalization Compatibility for SAS String Functions .............. 343

Chapter 12
VARCHAR Support in Functions .............................................. 371

Chapter 13
Dictionary of Functions for NLS .............................................. 375
Internationalization Compatibility for SAS String Functions

SAS provides string functions and CALL routines that enable you to easily manipulate your character data. Many of the original SAS string functions assume that the size of one character is always 1 byte. This process works well for data in a single-byte character set (SBCS). However, when some of these functions and CALL routines are used with data in a double-byte character set (DBCS) or a multi-byte character set (MBCS) such as UTF-8, the data is often handled improperly, and the string functions produce incorrect results.

To solve this problem SAS introduced a set of string functions and CALL routines, called K functions, for those string manipulations where DBCS or MBCS data must be handled carefully. The K functions do not make assumptions about the size of a character in a string. Table 11.4 on page 345 shows the level of I18N compatibility for each SAS string function. I18N is the abbreviation for internationalization. Compatibility indicates whether a program using a particular string function can be adapted to different languages and locales without program changes.

It is important to understand the difference between byte-based offset-length and character-based offset-length in order to use the K functions properly.

A byte-based offset assumes that the starting position specified for a character is the byte position of that character in the string. For SBCS data, where one character is always 1 byte in length, you can assume that the second character in the string begins in byte 2. However, if the data in the string is multi-byte data, the data in byte 2 might be one of the following, depending on the data and the encoding of the data:

- the second character in the string
- the second byte of a 2-byte character
- the first byte of the first character in the string.

A byte-based length represents the number of bytes in the string.
A character-based offset assumes that the position specified is the position of the character in the string. For all encodings, a character-based position of 2 is always the second character in the string. You cannot assume that you know the size of the characters in the string.

A character-based length represents the number of characters in the string.

K functions use a character-based offset or length, which does not take into consideration the byte position of the character in the string.

Here is an example of the functionality of a SAS string function and its corresponding K function.

In the UTF8 encoding, the STR='E282AC313233'x string has 6 bytes and represents four characters.

### Table 11.1  String Representation in UTF-8

<table>
<thead>
<tr>
<th>Hexadecimal Representation</th>
<th>E282AC</th>
<th>31</th>
<th>32</th>
<th>33</th>
</tr>
</thead>
<tbody>
<tr>
<td>€</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

The following example uses this same hexadecimal string in the SUBSTR function, which returns an unexpected value in UTF-8. The expected returned string would be the first character, the euro symbol. However, the result is an invalid character. The LENGTH function is byte-oriented, so it reads the individual bytes of data. The SUBSTR function returns the first byte of the first character, which is an incomplete value because the first character is encoded on 3 bytes.

### Table 11.2  Code Example Using Regular Functions

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>/* SAS program to submit in a UTF-8 SAS session */ data <em>null</em>;</td>
<td>str=E282AC313233</td>
</tr>
<tr>
<td>str= “€123” ;</td>
<td>s= s1=1 s=E22020202020</td>
</tr>
<tr>
<td>s=substr(str,1,1) ;</td>
<td>l=6</td>
</tr>
<tr>
<td>sl=length(s) ;</td>
<td></td>
</tr>
<tr>
<td>l=length(str) ;</td>
<td></td>
</tr>
<tr>
<td>put str= $hex16. /s= sl= / s= $hex. /l=;</td>
<td></td>
</tr>
<tr>
<td>run ;</td>
<td></td>
</tr>
</tbody>
</table>

The code in Table 11.3 on page 345 solves this issue by substituting the SUBSTR() function, which works only on single-byte data, with the KSUBSTR() function, which works on single and multi-byte data. Direct substitutions can also be made for other string functions, such as changing the LENGTH() or INDEX() functions to use the KLENGTH() or KINDEX() functions. Because the two functions read the data differently, you get different results when running this code in a multi-byte environment.
Before replacing all of the original SAS string-handling functions with K functions, examine your SAS program. If the string function processes data that contains only single-byte characters, K functions are not necessary. For example, strings containing XML tags do not require the use of K functions. Knowing the character data that is in your SAS programs and how it is processed can save unnecessary updates to your SAS code. The processing of binary data is not supported by the string-handling K functions, which expect strings to match the current session encoding.

String functions are assigned I18N levels depending on whether the functions can process DBCS, MBCS, or SBCS. Here are descriptions of the levels:

**I18N Level 0**
This function is designed for SBCS data. Do not use this function to process MBCS data.

**I18N Level 1**
This function should be avoided, if possible, if you are processing MBCS data. The I18N Level 1 functions might not work correctly with DBCS or MBCS encodings under certain circumstances.

**I18N Level 2**
This function can be used for SBCS, DBCS, and MBCS (UTF-8) data.

### Table 11.4 SAS String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>I18N Level 0</th>
<th>I18N Level 1</th>
<th>I18N Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ANORM420 Function” (p. 382)</td>
<td>Returns a normalized string from an input string encoded in EBCDIC420.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYALNUM Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for an alphanumeric character, and returns the first position at which the character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“ANYALPHA Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for an alphabetic character, and returns the first position at which the character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYCNTRL Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a control character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYDIGIT Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a digit, and returns the first position at which the digit is found. <strong>Note:</strong> This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“ANYFIRST Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is valid as the first character in a SAS variable name under VALIDVARNAME=V7, and returns the first position at which that character is found. Note: This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYGRAPH Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a graphical character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYLOWER Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a lowercase letter, and returns the first position at which the letter is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“ANYNAME Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is valid in a SAS variable name under VALIDVARNAME=V7, and returns the first position at which that character is found. Note: This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYPRINT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a printable character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYPUNCT Function” in SAS Functions and CALL Routines: Reference</td>
<td>searches a character string for a punctuation character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“ANYSPACE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a white-space-character (blank, horizontal and vertical tab, carriage return, line feed, and form feed). Returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYUPPER Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for an uppercase letter, and returns the first position at which the letter is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYXDIGIT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a hexadecimal character that represents a digit, and returns the first position at which that character is found. <strong>Note:</strong> This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“BASECHAR Function” (p. 392)</td>
<td>Converts characters to base characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“BYTE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns one character in the ASCII or the EBCDIC collating sequence.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“CAT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Does not remove leading or trailing blanks, and returns a concatenated character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“CATQ Function” in SAS Functions and CALL Routines: Reference</td>
<td>Concatenates character or numeric values by using a delimiter to separate items and by adding quotation marks to strings that contain the delimiter.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“CATS Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes leading and trailing blanks, and returns a concatenated character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“CATT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes trailing blanks, and returns a concatenated character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“CATX Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes leading and trailing blanks, inserts delimiters, and returns a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“CHAR Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a single character from a specified position in a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“CHOOSEC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character value that represents the results of choosing from a list of arguments.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“CHOOSEN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a numeric value that represents the results of choosing from a list of arguments.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“COALESCE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the first nonmissing value from a list of numeric arguments.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“COLLATE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character string in ASCII or EBCDIC collating sequence.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“COMPARE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the position of the leftmost character by which two strings differ, or returns 0 if there is no difference.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“COMPBL Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes multiple blanks from a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“COMPGED Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the generalized edit distance between two strings.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“COMPLEV Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the Levenshtein edit distance between two strings.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“COMPRESS Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character string with specified characters removed from the original string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“COUNT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Counts the number of times that a specified substring appears within a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“COUNTC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Counts the number of characters in a string that appear or do not appear in a list of characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“COUNTW Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Counts the number of words in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“DEQUOTE Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Removes matching quotation marks from a character string that begins with a quotation mark, and deletes all characters to the right of the closing quotation mark.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“FIND Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches for a specific substring of characters within a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“FINDC Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a string for any character in a list of characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“FINDW Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns the character position of a word in a string, or returns the number of the word in a string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“FIRST Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns the first character in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“GETLOCENV Function” (p. 397)</td>
<td>Returns the current locale/language environment.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLANGUAGE Function” (p. 398)</td>
<td>Returns the current two-letter language code.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLOCAL Function” (p. 399)</td>
<td>Returns the POSIX locale value for a SAS locale.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“GETPXREGION Function” (p. 400)</td>
<td>Returns the current two-letter region code.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“HTMLDECODE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Decodes a string that contains HTML numeric character references or HTML character entity references, and returns the decoded string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“HTMLENCODE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Encodes characters using HTML character entity references, and returns the encoded string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“IFC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character value based on whether an expression is true, false, or missing.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“IFN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a numeric value based on whether an expression is true, false, or missing.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“INDEX Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character expression for a string of characters, and returns the position of the string’s first character for the first occurrence of the string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“INDEXC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character expression for any of the specified characters, and returns the position of that character.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“INDEXW Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character expression for a string that is specified as a word, and returns the position of the first character in the word.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOMPARE Function” (p. 403)</td>
<td>Returns the result of a comparison of character expressions.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KCOMPRESS Function” (p. 405)</td>
<td>Removes specified characters from a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KCOUNT Function” (p. 407)</td>
<td>Returns the number of double-byte characters in an expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KCOUNTC Function” (p. 408)</td>
<td>counts individual characters in a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KCOUNTW Function” (p. 410)</td>
<td>Counts the number of words in a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KCOUNTX Function” (p. 412)</td>
<td>Counts the number of times that a specified substring appears within a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KCVT Function” (p. 414)</td>
<td>Converts data from one type of encoding data to another encoding data.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KFINX Function” (p. 416)</td>
<td>Searches for a specific substring of characters within a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KFINDC Function” (p. 418)</td>
<td>Searches a string for any character in a list of characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KFINDW Function” (p. 421)</td>
<td>Returns the character position of a word in a string or the number of the word in a string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KINDEX Function” (p. 425)</td>
<td>Searches a character expression for a string of characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KINDEXB Function” (p. 426)</td>
<td>Searches a character expression for a string of characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KINDEXCB Function” (p. 427)</td>
<td>Searches a character expression for specified characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KINDEXC Function” (p. 428)</td>
<td>Searches a character expression for specified characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KLEFT Function” (p. 430)</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO-SI.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KLENGTH Function” (p. 432)</td>
<td>Returns the length of an argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KLOWCASE Function” (p. 433)</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KPROPCASE Function” (p. 435)</td>
<td>Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KPROPCHAR Function” (p. 438)</td>
<td>Converts special characters to normal characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPDATA Function” (p. 439)</td>
<td>Removes or converts unprintable characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KREVERSE Function” (p. 443)</td>
<td>Reverses a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KRIGHT Function” (p. 444)</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO-SI.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSCAN Function” (p. 445)</td>
<td>Selects a specified word from a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSCANX Function” (p. 447)</td>
<td>Selects a specified word from a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSTRCAT Function” (p. 451)</td>
<td>Concatenates two or more character expressions.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSTRIP Function” (p. 452)</td>
<td>Removes leading and trailing blanks from a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSUBSTR Function” (p. 454)</td>
<td>Extracts a substring from an argument.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSUBSTRB Function” (p. 455)</td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KSUBSTRN Function” (p. 457)</td>
<td>Returns a substring, allowing a result with a length of zero.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KTRANSLATE Function” (p. 458)</td>
<td>Replaces specific characters in a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KTRIM Function” (p. 460)</td>
<td>Removes trailing DBCS blanks and SO-SI from character expressions.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KTRUNCATE Function” (p. 462)</td>
<td>Truncates a character string to a specified length in byte units without breaking multibyte characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KUPCASE Function” (p. 463)</td>
<td>Converts all letters in an argument to uppercase.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KUPDATE Function” (p. 464)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KUPDATEB Function” (p. 466)</td>
<td>Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KUPDATES Function” (p. 468)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KVERIFYB Function” (p. 471)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KVERIFY Function” (p. 470)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LEFT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Left-aligns a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LENGTH Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the length of a non-blank character string, excluding trailing blanks, and returns 1 for a blank character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LENGTHC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, including trailing blanks.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LENGTHM Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the amount of memory (in bytes) that is allocated for a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LENGTHN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, excluding trailing blanks.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LOWCASE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“MD5 Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the result of the message digest of a specified string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“MISSING Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a numeric result that indicates whether the argument contains a missing value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“MVALID Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Checks the validity of a character string for use as a SAS member name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLDATE Function” (p. 472)</td>
<td>Converts the SAS date value to the date value of the specified locale by using the date format descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLDATM Function” (p. 475)</td>
<td>Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLTIME Function” (p. 478)</td>
<td>Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLITERAL Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Converts a character string that you specify to a SAS name literal.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTALNUM Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a non-alphanumeric character, and returns the first position at which the character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“NOTALPHA Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a nonalphabetic character, and returns the first position at which the character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“NOTCNTRL Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a character that is not a control character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
| “NOTDIGIT Function” in *SAS Functions and CALL Routines: Reference* | Searches a character string for any character that is not a digit, and returns the first position at which that character is found.  
*Note:* This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status. |  | X |  |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>I18N Level 0</th>
<th>I18N Level 1</th>
<th>I18N Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>“NOTFIRST Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for an invalid first character in a SAS variable name under VALIDVARNAME= V7, and returns the first position at which that character is found. Note: This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NOTGRAPH Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a non-graphical character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“NOTLOWER Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a lowercase letter, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“NOTNAME Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for an invalid character in a SAS variable name under VALIDVARNAME= V7, and returns the first position at which that character is found. Note: This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“NOTPRINT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a nonprintable character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“NOTPUNCT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a punctuation character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“NOTSPACE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a white-space-character (blank, horizontal and vertical tab, carriage return, line feed, and form feed), and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“NOTUPPER Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not an uppercase letter, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“NOTXDIGIT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a hexadecimal character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Note: This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NVALID Function” in SAS Functions and CALL Routines: Reference</td>
<td>Checks the validity of a character string for use as a SAS variable name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“PROPCASE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Converts all words in an argument to proper case.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“PRXCHANGE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Performs a pattern-matching replacement.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“PRXMATCH Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches for a pattern match and returns the position at which the pattern is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“PRXPAREN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the last bracket match for which there is a match in a pattern.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“PRXPARSE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Compiles a Perl regular expression (PRX) that can be used for pattern matching of a character value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“PRXPOSN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character string that contains the value for a capture buffer.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“PUT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a value using a specified format.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“PUTC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Enables you to specify a character format at run time.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“PUTN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Enables you to specify a numeric format at run time.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“QUOTE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Adds double quotation marks to a character value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“RANK Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the position of a character in the ASCII or EBCDIC collating sequence.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“REPEAT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character value that consists of the first argument repeated n+1 times.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“REVERSE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Reverses a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“RIGHT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Right-aligns a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SASMSG Function” (p. 479)</td>
<td>Specifies a message from a data set. The returned message is based on the current locale and a specified key.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SASMSGL Function” (p. 482)</td>
<td>Specifies a message from a data set. The message is based on a specified locale value and a specified key value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SCAN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the nth word from a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SETLOCALE Function” (p. 485)</td>
<td>Specifies the locale keys for the current SAS locale.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SHA256 Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the result of the message digest of a specified string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SORTKEY Function” (p. 493)</td>
<td>Creates a linguistic sort key.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SOUNDEX Function” in SAS Functions and CALL Routines: Reference</td>
<td>Encodes a string to facilitate searching.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“SPEDIS Function” in SAS Functions and CALL Routines: Reference</td>
<td>Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“STRIP Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character string with all leading and trailing blanks removed.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“SUBPAD Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a substring that has a length that you specify, using blank padding if necessary.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“SUBSTR (left of =) Function” in SAS Functions and CALL Routines: Reference</td>
<td>Replaces character value contents</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“SUBSTR (right of =) Function” in SAS Functions and CALL Routines: Reference</td>
<td>Extracts a substring from an argument.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“SUBSTRN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a substring, allowing a result with a length of zero.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“TRANSLATE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Replaces specific characters in a character string.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“TRANSTRN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Replaces or removes all occurrences of a substring in a character string.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“TRANWRD Function” on page 505</td>
<td>Transcodes data by using the specified translation table.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>“TRANTAB Function” on page 505</td>
<td>Replaces or removes all occurrences of a substring in a character string.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>&quot;TRIM Function&quot; in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Removes trailing blanks from a character string, and returns one blank if the string is missing.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;TRIMN Function&quot; in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Removes trailing blanks from character expressions, and returns a string with a length of zero if the expression is missing.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;TZONEDSTNAME Function&quot; (p. 501)</td>
<td>Returns a daylight savings time name.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>&quot;TZONEDSTOFF Function&quot; (p. 502)</td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>&quot;TZONEDSTNAME Function&quot; (p. 503)</td>
<td>Returns a standard time zone name.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>&quot;TZONEDSTOFF Function&quot; (p. 504)</td>
<td>Returns the time zone offset value for the specified standard time.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“TZON EU2S Function” (p. 504)</td>
<td>Converts a UTC date time value to a SAS date time value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODE Function” (p. 507)</td>
<td>Converts Unicode characters to the current SAS session encoding.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODEDEC Function” (p. 509)</td>
<td>Converts characters in the current SAS session encoding to Unicode characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODELEN Function” (p. 510)</td>
<td>Specifies the length of the character unit for the Unicode data.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODEWIDTH Function” (p. 511)</td>
<td>Specifies the length of a display unit for the Unicode data.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UPCASE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“URLDECODE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a string that was decoded using the URL escape syntax.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“URLENCODE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a string that was encoded using the URL escape syntax.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“VERIFY Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the position of the first character in a string that is not in any of several other strings.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>&quot;WHICHC Function&quot; in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches for a character value that is equal to the first argument, and returns the index of the first matching value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
SAS supports the VARCHAR data type for character-data processing in the DATA step and PROC step. The CAS engine is required to read or write VARCHAR data in order to preserve the variable’s type. Variables created using the VARCHAR data type vary in width and use character semantics, rather than being fixed-width and using byte semantics of the original SAS character data type. When a VARCHAR variable is passed to any string function, including the traditional string functions or the K functions, SAS assumes that the length represents the number of characters. An offset for a VARCHAR variable represents the position of a character in the string. The fixed-width character data type is better suited for binary data.

VARCHAR is supported in CAS tables and is used in a data_NULL_step. Some procedures support VARCHAR columns. Procedures that do not support VARCHAR columns convert the columns to a fixed-width character column. VARCHAR is not supported in data sets.

Here is an example of the data types using the SUBSTR function: $\text{substr}(X, 10, 2)$.

If the variable X is defined as a fixed-width character variable, SUBSTR returns the 10th and 11th bytes in the string. If X is defined as a VARCHAR variable, SUBSTR returns the 10th and 11th characters in the string.

A variable that is assigned a VARCHAR data type of length 10 can hold up to 10 characters.

This table shows the differences between fixed-width and VARCHAR data types. The fixed-width character data type uses byte semantics and VARCHAR uses character semantics.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Traditional String Function</th>
<th>K Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed-width character</td>
<td>byte</td>
<td>character</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>character</td>
<td>character</td>
</tr>
</tbody>
</table>
The SAS Viya session encoding, UTF-8, supports data that can be 1 or multiple bytes. An example of a multi-byte UTF-8 character is the euro (€) character, which has 3 bytes. Here is a table that shows the representation of the string €123. The € character has 3 bytes and each number has 1 byte.

<table>
<thead>
<tr>
<th>Character</th>
<th>€</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexadecim al value</td>
<td>E2</td>
<td>82</td>
<td>AC</td>
<td>31</td>
</tr>
</tbody>
</table>

The following examples compare the fixed-width data type and the VARCHAR data type. Table 12.1 on page 372 shows results from the fixed-width variable using the data €123. Table 12.2 on page 373 shows results using VARCHAR.

K functions should be used for fixed-width character data that might contain multi-byte characters:

1. The example in the first row assigns the string to a fixed-width character column. The PUT statement displays the characters and then displays the hexadecimal value for those characters.

2. The example in the second row uses the SUBSTR function with a position of 2. Because the variable x is a fixed-width character column, the position interpreted as the second byte of the string in x, which is also the second byte of the € character. The result is that the first 2 bytes assigned to xsub are invalid UTF-8 data. The results displayed here show garbage (or the Unicode replacement character) as the first 2 bytes of xsub.

3. The example in the third row uses the INDEX function to locate the 1 in the string. Because the variable x is a fixed-width character column, the result returned by the INDEX function is the byte position of the character in the string. The result is xidx=4 because the 1 is located at the fourth byte in x.

K functions should be used for fixed-width character data that might contain multi-byte characters:

Table 12.1 Fixed-Width Character Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>x=€123 x=E282AC</td>
</tr>
<tr>
<td>x='€123';</td>
<td>xsub=123 xsub=82AC31323320</td>
</tr>
<tr>
<td>put x=</td>
<td>xidx=4</td>
</tr>
<tr>
<td>x= $hex6.;</td>
<td></td>
</tr>
<tr>
<td>xsub=substr(x,2);</td>
<td></td>
</tr>
<tr>
<td>put xsub=</td>
<td></td>
</tr>
<tr>
<td>xsub= $hex.;</td>
<td></td>
</tr>
<tr>
<td>xidx=index(x,'1');</td>
<td></td>
</tr>
<tr>
<td>put xidx=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

1. The example in the first row assigns the string to a VARCHAR column. The PUT statement displays the characters and then the hexadecimal value for those characters.

2. The example in the second row uses the SUBSTR function with a position of 2. In this example, the variable v is defined as a VARCHAR so that the position is
interpreted as the second character of the string in v. The result assigned to vsub is the string '123', which contains valid UTF-8 characters.

3 The example in the third row uses the INDEX function to locate the 1 in the string. Because the variable v is a VARCHAR column, the result returned by the INDEX function is the character position of the character in the string. The result is vidx=2 because 1 is the second character in v.

Table 12.2  VARCHAR Character Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>v=€$$VARIABLE$$3 $hex.;</td>
</tr>
<tr>
<td>length v VARCHAR(6);</td>
<td>vsub=3132332020202020202020202020202020</td>
</tr>
<tr>
<td>v='123';</td>
<td>vidx=2</td>
</tr>
<tr>
<td>put v=</td>
<td></td>
</tr>
<tr>
<td>v= $hex.;</td>
<td></td>
</tr>
<tr>
<td>vsub=substr(v,2);</td>
<td></td>
</tr>
<tr>
<td>put vsub=</td>
<td></td>
</tr>
<tr>
<td>vsub= $hex.;</td>
<td></td>
</tr>
<tr>
<td>vidx=index(v,'1');</td>
<td></td>
</tr>
<tr>
<td>put vidx=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

VARCHAR Data Type in String Functions 373
Dictionary of Functions for NLS

Functions by Category

Dictionary

ANORM420 Function ........................................ 382
CALL KSCANX Routine ...................................... 382
BASECHAR Function ......................................... 387
ENCODCOMPAT Function .................................... 392
ENCODISVALID Function .................................... 394
GETLOCEVFunction .......................................... 396
GETPXENV Function .......................................... 397
GETPXLANGUAGE Function .................................... 398
GETPXLOCALE Function ...................................... 399
GETPXREGION Function ...................................... 400
KCHARLIST Function .......................................... 401
KCOMPARE Function .......................................... 403
KCHARLIST Function .......................................... 405
KCOUNT Function ........................................... 407
KCOUNTC Function ........................................... 408
KCOUNTW Function ........................................... 410
KCOUNTX Function ........................................... 412
KCVT Function ................................................ 414
KFINDD Function ............................................. 416
KFINDC Function ............................................. 418
KFINDW Function ............................................. 421
KINDEX Function ............................................. 425
KINDEXB Function ............................................ 426
KINDEXCB Function .......................................... 427
KINDEXC Function ............................................ 428
KLEFT Function ............................................... 430
KLENGTH Function ............................................ 432
KLOWCASE Function .......................................... 433
KPROPBCASE Function ....................................... 435
KPROPCHAR Function ........................................ 438
KPROPDATA Function ........................................ 439
KREVERSE Function .......................................... 443
KRIGHT Function ............................................. 444
KSCAN Function .............................................. 445
KSCANX Function ............................................. 447
KSTRCAT Function ............................................ 451
KSTRIP Function ............................................... 452
KSUBSTR Function ............................................ 454
KSUBSTRB Function .......................................... 455
KSUBSTRN Function ........................................... 457
KTRANSLATE Function ....................................... 458
Functions by Category

Some functions run in SAS only, and some functions run in SAS and on the CAS engine. If CAS is specified for the function category, then the function runs in SAS and on the CAS server. If CAS is not specified for the function category, then the function runs in SAS only. For example, the BASECHAR function runs in SAS and on the CAS server, so CAS is specified as a category. The KCVT function runs on SAS only, so CAS is not specified as a category.

The following categories relate to NLS issues:

<table>
<thead>
<tr>
<th>Table 13.1</th>
<th>Categories of NLS Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>CAS</td>
<td>Functions that run on the CAS server.</td>
</tr>
<tr>
<td>Character</td>
<td>processes character data</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Currency Conversion</td>
<td>converts one currency to another currency</td>
</tr>
<tr>
<td>DBCS</td>
<td>processes double-byte character set.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>processes data and time data.</td>
</tr>
<tr>
<td>Locale</td>
<td>processes data based on the specified locale.</td>
</tr>
<tr>
<td>Variable Information</td>
<td>processes variable information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td>CALL KSCANX Routine (p. 387)</td>
<td>Returns the position and length of the nth word from a character string.</td>
</tr>
<tr>
<td></td>
<td>BASECHAR Function (p. 392)</td>
<td>Converts characters to base characters.</td>
</tr>
<tr>
<td></td>
<td>KCHARLIST Function (p. 401)</td>
<td>Returns the character list according to the modifier.</td>
</tr>
<tr>
<td></td>
<td>KCOMPARE Function (p. 403)</td>
<td>Returns the result of a comparison of character expressions.</td>
</tr>
<tr>
<td></td>
<td>KCOMPRESS Function (p. 405)</td>
<td>Removes specified characters from a character expression.</td>
</tr>
<tr>
<td></td>
<td>KCOUNT Function (p. 407)</td>
<td>Returns the number of double-byte characters in an expression.</td>
</tr>
<tr>
<td></td>
<td>KCOUNTC Function (p. 408)</td>
<td>Counts individual characters in a character string.</td>
</tr>
<tr>
<td></td>
<td>KCOUNTW Function (p. 410)</td>
<td>Counts the number of words in a character string.</td>
</tr>
<tr>
<td></td>
<td>KCOUNTX Function (p. 412)</td>
<td>Counts the number of times that a specified substring appears within a character string.</td>
</tr>
<tr>
<td></td>
<td>KCVT Function (p. 414)</td>
<td>Converts data from one type of encoding data to another type of encoding data.</td>
</tr>
<tr>
<td></td>
<td>KFIND Function (p. 416)</td>
<td>Searches for a specific substring of characters within a character string.</td>
</tr>
<tr>
<td></td>
<td>KFINDC Function (p. 418)</td>
<td>Searches a string for any character in a list of characters.</td>
</tr>
<tr>
<td></td>
<td>KFINDW Function (p. 421)</td>
<td>Returns the character position of a word in a string or the number of the word in a string.</td>
</tr>
<tr>
<td></td>
<td>KINDEX Function (p. 425)</td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td></td>
<td>KINDEXC Function (p. 428)</td>
<td>Searches a character expression for specified characters and returns character-based values.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>KLEFT Function (p. 430)</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td></td>
<td>KLENGTH Function (p. 432)</td>
<td>Returns the length of an argument.</td>
</tr>
<tr>
<td></td>
<td>KLOWCASE Function (p. 433)</td>
<td>Converts the uppercase alphabetic letters to lowercase letters.</td>
</tr>
<tr>
<td></td>
<td>KREVERSE Function (p. 443)</td>
<td>Reverses a character expression.</td>
</tr>
<tr>
<td></td>
<td>KRIGHT Function (p. 444)</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td></td>
<td>KSCAN Function (p. 445)</td>
<td>Selects a specified word from a character expression.</td>
</tr>
<tr>
<td></td>
<td>KSCANX Function (p. 447)</td>
<td>Selects a specified word from a character expression using a modifier to process the function's action.</td>
</tr>
<tr>
<td></td>
<td>KSTRCAT Function (p. 451)</td>
<td>Concatenates two or more character expressions.</td>
</tr>
<tr>
<td></td>
<td>KSTRIP Function (p. 452)</td>
<td>Removes leading and trailing blanks from a character string.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTR Function (p. 454)</td>
<td>Extracts a substring from an argument.</td>
</tr>
<tr>
<td></td>
<td>KTRANSLATE Function (p. 458)</td>
<td>Replaces specific characters in a character expression.</td>
</tr>
<tr>
<td></td>
<td>KTRIM Function (p. 460)</td>
<td>Removes trailing DBCS blanks and SO/SI from character expressions.</td>
</tr>
<tr>
<td></td>
<td>KUPCASE Function (p. 463)</td>
<td>Converts the lowercase alphabetic letters to uppercase letters.</td>
</tr>
<tr>
<td></td>
<td>KUPDATE Function (p. 464)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td></td>
<td>KUPDATES Function (p. 468)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td></td>
<td>KVERIFY Function (p. 470)</td>
<td>Returns the position of the first character (character-based value) that is unique to an expression.</td>
</tr>
<tr>
<td></td>
<td>VTRANSCODE Function (p. 514)</td>
<td>Returns a value that indicates whether transcoding is enabled for the specified character variable.</td>
</tr>
<tr>
<td></td>
<td>VTRANSCODEX Function (p. 515)</td>
<td>Returns a value that indicates whether transcoding is enabled for the specified argument.</td>
</tr>
<tr>
<td></td>
<td>ANORM420 Function (p. 382)</td>
<td>Returns a normalized string from an input string encoded in EBCDIC420.</td>
</tr>
<tr>
<td></td>
<td>BASECHAR Function (p. 392)</td>
<td>Converts characters to base characters.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Language Elements</td>
<td>KCVT Function (p. 414)</td>
<td>Converts data from one type of encoding data to another type of encoding data.</td>
</tr>
<tr>
<td></td>
<td>KSTRIP Function (p. 452)</td>
<td>Removes leading and trailing blanks from a character string.</td>
</tr>
<tr>
<td></td>
<td>TRANTAB Function (p. 505)</td>
<td>Transcodes data by using the specified translation table.</td>
</tr>
<tr>
<td></td>
<td>UNICODE Function (p. 507)</td>
<td>Converts Unicode characters to the current SAS session encoding.</td>
</tr>
<tr>
<td></td>
<td>UNICODEC Function (p. 509)</td>
<td>Converts characters in the current SAS session encoding to Unicode characters.</td>
</tr>
<tr>
<td></td>
<td>UNICODELEN Function (p. 510)</td>
<td>Specifies the length of the character unit for the Unicode data.</td>
</tr>
<tr>
<td></td>
<td>UNICODEWIDTH Function (p. 511)</td>
<td>Specifies the length of a display unit for the Unicode data.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>NLDATE Function (p. 472)</td>
<td>Converts the SAS date value to the date value of the specified locale by using the date format descriptors.</td>
</tr>
<tr>
<td></td>
<td>NLDATM Function (p. 475)</td>
<td>Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.</td>
</tr>
<tr>
<td></td>
<td>NLTIME Function (p. 478)</td>
<td>Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.</td>
</tr>
<tr>
<td></td>
<td>TZONEID Function (p. 496)</td>
<td>Returns the current time zone ID.</td>
</tr>
<tr>
<td></td>
<td>TZONENAME Function (p. 497)</td>
<td>Returns the current standard or daylight savings time, time zone name.</td>
</tr>
<tr>
<td></td>
<td>TZONEOFF Function (p. 498)</td>
<td>Returns the user time zone offset.</td>
</tr>
<tr>
<td></td>
<td>TZONE2U Function (p. 500)</td>
<td>Converts a SAS date time value to a UTC date time value.</td>
</tr>
<tr>
<td></td>
<td>TZONE2STNAME Function (p. 501)</td>
<td>Returns a daylight savings time name.</td>
</tr>
<tr>
<td></td>
<td>TZONE2STOFF Function (p. 502)</td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
</tr>
<tr>
<td></td>
<td>TZONESTTNAME Function (p. 503)</td>
<td>Returns a standard time zone name.</td>
</tr>
<tr>
<td></td>
<td>TZONESTTOFF Function (p. 504)</td>
<td>Returns the time zone offset value for the specified standard time.</td>
</tr>
<tr>
<td></td>
<td>TZONEU2S Function (p. 504)</td>
<td>Converts a UTC date time value to a SAS date time value.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>DBCS</td>
<td>CALL KSCANX Routine (p. 387)</td>
<td>Returns the position and length of the nth word from a character string.</td>
</tr>
<tr>
<td></td>
<td>KCHARLIST Function (p. 401)</td>
<td>Returns the character list according to the modifier.</td>
</tr>
<tr>
<td></td>
<td>KCOMPARE Function (p. 403)</td>
<td>Returns the result of a comparison of character expressions.</td>
</tr>
<tr>
<td></td>
<td>KCOMPRESS Function (p. 405)</td>
<td>Removes specified characters from a character expression.</td>
</tr>
<tr>
<td></td>
<td>KCOUNT Function (p. 407)</td>
<td>Returns the number of double-byte characters in an expression.</td>
</tr>
<tr>
<td></td>
<td>KCOUNTC Function (p. 408)</td>
<td>Counts individual characters in a character string.</td>
</tr>
<tr>
<td></td>
<td>KCOUNTW Function (p. 410)</td>
<td>Counts the number of words in a character string.</td>
</tr>
<tr>
<td></td>
<td>KCOUNTX Function (p. 412)</td>
<td>Counts the number of times that a specified substring appears within a character string.</td>
</tr>
<tr>
<td></td>
<td>KFIND Function (p. 416)</td>
<td>Searches for a specific substring of characters within a character string.</td>
</tr>
<tr>
<td></td>
<td>KFINDC Function (p. 418)</td>
<td>Searches a string for any character in a list of characters.</td>
</tr>
<tr>
<td></td>
<td>KFINDW Function (p. 421)</td>
<td>Returns the character position of a word in a string or the number of the word in a string.</td>
</tr>
<tr>
<td></td>
<td>KINDEX Function (p. 425)</td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td></td>
<td>KINDEXB Function (p. 426)</td>
<td>Searches a character expression for specified characters and returns byte-based values.</td>
</tr>
<tr>
<td></td>
<td>KINDEXCB Function (p. 427)</td>
<td>Searches a character expression for specified characters and returns byte-based values.</td>
</tr>
<tr>
<td></td>
<td>KINDEXC Function (p. 428)</td>
<td>Searches a character expression for specified characters and returns character-based values.</td>
</tr>
<tr>
<td></td>
<td>KLEFT Function (p. 430)</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td></td>
<td>KLENGTH Function (p. 432)</td>
<td>Returns the length of an argument.</td>
</tr>
<tr>
<td></td>
<td>KLOWCASE Function (p. 433)</td>
<td>Converts the uppercase alphabetic letters to lowercase letters.</td>
</tr>
<tr>
<td></td>
<td>KPROPCHAR Function (p. 438)</td>
<td>Converts Unicode to the corresponding characters enclosed in parenthesis.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>KPROPDATA Function (p. 439)</td>
<td>Removes or converts unprintable characters.</td>
</tr>
<tr>
<td></td>
<td>KREVERSE Function (p. 443)</td>
<td>Reverses a character expression.</td>
</tr>
<tr>
<td></td>
<td>KRIGHT Function (p. 444)</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td></td>
<td>KSCAN Function (p. 445)</td>
<td>Selects a specified word from a character expression.</td>
</tr>
<tr>
<td></td>
<td>KSCANX Function (p. 447)</td>
<td>Selects a specified word from a character expression using a modifier to process the function's action.</td>
</tr>
<tr>
<td></td>
<td>KSTRCAT Function (p. 451)</td>
<td>Concatenates two or more character expressions.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTR Function (p. 454)</td>
<td>Extracts a substring from an argument.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTRB Function (p. 455)</td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTRN Function (p. 457)</td>
<td>Returns a substring, allowing a result with a length of zero.</td>
</tr>
<tr>
<td></td>
<td>KTRANSLATE Function (p. 458)</td>
<td>Replaces specific characters in a character expression.</td>
</tr>
<tr>
<td></td>
<td>KTRIM Function (p. 460)</td>
<td>Removes trailing DBCS blanks and SO/SI from character expressions.</td>
</tr>
<tr>
<td></td>
<td>KTRUNCATE Function (p. 462)</td>
<td>Truncates a string to a specified length in byte unit without breaking multibyte characters.</td>
</tr>
<tr>
<td></td>
<td>KUPCASE Function (p. 463)</td>
<td>Converts the lowercase alphabetic letters to uppercase letters.</td>
</tr>
<tr>
<td></td>
<td>KUPDATE Function (p. 464)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td></td>
<td>KUPDATEB Function (p. 466)</td>
<td>Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.</td>
</tr>
<tr>
<td></td>
<td>KUPDATES Function (p. 468)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td></td>
<td>KVERIFY Function (p. 470)</td>
<td>Returns the position of the first character (character-based value) that is unique to an expression.</td>
</tr>
<tr>
<td></td>
<td>KVERIFYB Function (p. 471)</td>
<td>Returns the position of the first character (byte-based value) that is unique to an expression.</td>
</tr>
</tbody>
</table>

Encoding | ENCODCOMPAT Function (p. 394) | Verifies the transcoding compatibility between two encodings.                                        |
<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENCODISVALID Function (p. 396)</td>
<td>Verifies a valid encoding name.</td>
<td></td>
</tr>
<tr>
<td>Locale</td>
<td>GETLOCENV Function (p. 397)</td>
<td>Returns the current locale/language environment.</td>
</tr>
<tr>
<td></td>
<td>GETPXLANGUAGE Function (p. 398)</td>
<td>Returns the current two-letter language code.</td>
</tr>
<tr>
<td></td>
<td>GETPXLOCALE Function (p. 399)</td>
<td>Returns the POSIX locale value for a SAS locale.</td>
</tr>
<tr>
<td></td>
<td>GETPXREGION Function (p. 400)</td>
<td>Returns the current two-letter region code.</td>
</tr>
<tr>
<td></td>
<td>SASMSG Function (p. 479)</td>
<td>Specifies a message from a data set. The returned message is based on the current locale and a specified key.</td>
</tr>
<tr>
<td></td>
<td>SASMSGL Function (p. 482)</td>
<td>Specifies a message from a data set. The message is based on a specified locale value and a specified key value.</td>
</tr>
<tr>
<td></td>
<td>SETLOCALE Function (p. 485)</td>
<td>Specifies the locale keys for the current SAS locale.</td>
</tr>
<tr>
<td></td>
<td>SORTKEY Function (p. 493)</td>
<td>Creates a linguistic sort key.</td>
</tr>
<tr>
<td>Variable Information</td>
<td>VARTRANSCODE Function (p. 512)</td>
<td>Returns the transcode attribute of a SAS data set variable.</td>
</tr>
<tr>
<td></td>
<td>VTRANSCODE Function (p. 514)</td>
<td>Returns a value that indicates whether transcoding is enabled for the specified character variable.</td>
</tr>
<tr>
<td></td>
<td>VTRANSCODEX Function (p. 515)</td>
<td>Returns a value that indicates whether transcoding is enabled for the specified argument.</td>
</tr>
</tbody>
</table>

**Dictionary**

**ANORM420 Function**

Returns a normalized string from an input string encoded in EBCDIC420.

**Category:** Character

**Restriction:** This function is assigned an I18N Level 1 status. If possible, avoid I18N Level 1 functions if you are using a non-English language. Under certain circumstances, the I18N Level 1 functions might not work correctly with Double-Byte Character Set (DBCS) or Multi-Byte Character Set (MBCS) encodings. For more information, see [Internationalization Compatibility on page 343](#).
Syntax

**ANORM420** *(string,<modifiers>)*

Required Argument

*string*

specifies an input string that is encoded in EBCDIC420.

Optional Argument

*modifiers*

are character constants, variables, or expressions that specify one or more modifiers. The following modifiers can be in uppercase or lowercase:

- **g**: ignores the ligature
- **i**: ignores the Arabic-Indic numbers
- **p**: ignores mapping of shaped characters
- **s**: ignores the addition of space
- **t**: ignores transcoding
- **z**: ignores mapping of the zero-length-space character

Details

The ANORM420 function processes data that is encoded in EBCDIC420.

Refer to the following tables for information about the modifier variables:

The ANORM420 function replaces the following character and code points with the appropriate ligature unless the modifier *g* is specified.

**Table 13.2 Modifier g: Arabic Ligatures**

<table>
<thead>
<tr>
<th>Character</th>
<th>Unicode Value</th>
<th>From</th>
<th>To</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ﻝ</td>
<td>(U+FEF5)</td>
<td>0xB2</td>
<td>0xB1 + 0x47</td>
<td>Arabic Ligature Lam with Aleph with Madda above</td>
</tr>
<tr>
<td>ﻝ</td>
<td>(U+FEF6)</td>
<td>0xB3</td>
<td>0xB1 + 0x47</td>
<td>Arabic Ligature Lam with Aleph with Madda above</td>
</tr>
<tr>
<td>ﻝ</td>
<td>(U+FEF7)</td>
<td>0xB4</td>
<td>0xB1 + 0x49</td>
<td>Arabic Ligature Lam with Aleph with Hamza above</td>
</tr>
<tr>
<td>ﻝ</td>
<td>(U+FEF8)</td>
<td>0xB5</td>
<td>0xB1 + 0x49</td>
<td>Arabic Ligature Lam with Aleph with Hamza above</td>
</tr>
<tr>
<td>ﻝ</td>
<td>(U+FEFC)</td>
<td>0xB8</td>
<td>0xB1 + 0x56</td>
<td>Arabic Ligature Lam with Aleph</td>
</tr>
</tbody>
</table>
The ANORM420 function remaps the Arabic-Indic numbers to the digit unless the modifier $i$ is specified and no transcoding occurs.

**Table 13.3  Modifier $i$: Arabic-Indic Numbers**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>٠ U+0660</td>
<td>0x0F0</td>
</tr>
<tr>
<td>١ U+0661</td>
<td>0x0F1</td>
</tr>
<tr>
<td>٢ U+0662</td>
<td>0x0F2</td>
</tr>
<tr>
<td>٣ U+0663</td>
<td>0x0F3</td>
</tr>
<tr>
<td>٤ U+0664</td>
<td>0x0F4</td>
</tr>
<tr>
<td>٥ U+0665</td>
<td>0x0F5</td>
</tr>
<tr>
<td>٦ U+0666</td>
<td>0x0F6</td>
</tr>
<tr>
<td>٧ U+0667</td>
<td>0x0F7</td>
</tr>
<tr>
<td>٨ U+0668</td>
<td>0x0F8</td>
</tr>
<tr>
<td>٩ U+0669</td>
<td>0x0F9</td>
</tr>
</tbody>
</table>

The ANORM420 function remaps the shaped characters to their unshaped equivalent unless the modifier $p$ is specified.

**Table 13.4  Modifier $p$: Shaped Characters Mapping**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x43</td>
<td>U+FE7D</td>
</tr>
<tr>
<td>0x48</td>
<td>FE82</td>
</tr>
<tr>
<td>0x51</td>
<td>U+FE84</td>
</tr>
<tr>
<td>0x57</td>
<td>U+FE8E</td>
</tr>
<tr>
<td>0x59</td>
<td>U+FE91</td>
</tr>
<tr>
<td>0x64</td>
<td>U+FE97</td>
</tr>
<tr>
<td>0x66</td>
<td>U+FE9B</td>
</tr>
<tr>
<td>0x68</td>
<td>U+FE9F</td>
</tr>
<tr>
<td>0x70</td>
<td>U+FEA3</td>
</tr>
<tr>
<td>0x72</td>
<td>U+FEA7</td>
</tr>
</tbody>
</table>
The ANORM420 function adds a space after the following characters unless the modifier is specified.

**Table 13.5**  
*Modifier s: ignores the addition of a space*

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x78</td>
<td>U+FEB3</td>
</tr>
<tr>
<td>0x8A</td>
<td>U+FEB7</td>
</tr>
<tr>
<td>0x8C</td>
<td>U+FEBB</td>
</tr>
<tr>
<td>0x8E</td>
<td>U+FEBF</td>
</tr>
<tr>
<td>0x9B</td>
<td>U+FECA</td>
</tr>
<tr>
<td>0x9C</td>
<td>U+FECB</td>
</tr>
<tr>
<td>0x9D</td>
<td>U+FECC</td>
</tr>
<tr>
<td>0x9F</td>
<td>U+FECE</td>
</tr>
<tr>
<td>0xA0</td>
<td>U+FECF</td>
</tr>
<tr>
<td>0xAA</td>
<td>U+FED0</td>
</tr>
<tr>
<td>0xAC</td>
<td>U+FED3</td>
</tr>
<tr>
<td>0xAE</td>
<td>U+FED7</td>
</tr>
<tr>
<td>0xB0</td>
<td>U+FEDB</td>
</tr>
<tr>
<td>0xBA</td>
<td>U+FEDF</td>
</tr>
<tr>
<td>0xBC</td>
<td>U+FEE3</td>
</tr>
<tr>
<td>0xBE</td>
<td>U+FEE7</td>
</tr>
<tr>
<td>0xCB</td>
<td>U+FEEB</td>
</tr>
<tr>
<td>0xCD</td>
<td>U+FEEC</td>
</tr>
<tr>
<td>0xDB</td>
<td>U+FEF0</td>
</tr>
<tr>
<td>0xDD</td>
<td>U+FEF2</td>
</tr>
<tr>
<td>0xDE</td>
<td>U+FEF3</td>
</tr>
</tbody>
</table>

0x58 (U+0628)  Arabic Letter BEH
<table>
<thead>
<tr>
<th>Arabic Letter</th>
<th>Unicode Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEH</td>
<td>0x62</td>
<td>(U+0629) MARBUTA</td>
</tr>
<tr>
<td>THEH</td>
<td>0x65</td>
<td>(U+062B) Arabic Letter THEH</td>
</tr>
<tr>
<td>JEEM</td>
<td>0x67</td>
<td>(U+062C) Arabic Letter JEEM</td>
</tr>
<tr>
<td>HAH</td>
<td>0x69</td>
<td>(U+062D) Arabic Letter HAH</td>
</tr>
<tr>
<td>KH AH</td>
<td>0x71</td>
<td>(U+062E) Arabian Letter KH AH</td>
</tr>
<tr>
<td>SEEN</td>
<td>0x77</td>
<td>(U+0633) Arabic Letter SEEN</td>
</tr>
<tr>
<td>SHEEN</td>
<td>0x80</td>
<td>(U+0634) Arabic Letter SHEEN</td>
</tr>
<tr>
<td>SAD</td>
<td>0x8B</td>
<td>(U+0635) Arabic Letter SAD</td>
</tr>
<tr>
<td>DAD</td>
<td>0x8D</td>
<td>(U+0636) Arabic Letter DAD</td>
</tr>
<tr>
<td>AIN</td>
<td>0x9A</td>
<td>(U+0639) Arabic Letter AIN</td>
</tr>
<tr>
<td>AIN final form</td>
<td>0x9B</td>
<td>(U+FECA) Arabic Letter AIN final form</td>
</tr>
<tr>
<td>GHAIN</td>
<td>0x9E</td>
<td>(U+063A) Arabic Letter GHAIN</td>
</tr>
<tr>
<td>GHAIN final form</td>
<td>0x9F</td>
<td>(U+FECE) Arabic Letter GHAIN final form</td>
</tr>
<tr>
<td>FEH</td>
<td>0xAB</td>
<td>(U+0641) Arabic Letter FEH</td>
</tr>
<tr>
<td>QAF</td>
<td>0xAD</td>
<td>(U+0642) Arabic Letter QAF</td>
</tr>
<tr>
<td>KAF</td>
<td>0xAF</td>
<td>(U+0643) Arabic Letter KAF</td>
</tr>
<tr>
<td>LAM</td>
<td>0xB1</td>
<td>(U+0644) Arabic Letter LAM</td>
</tr>
<tr>
<td>MEEM</td>
<td>0xBB</td>
<td>(U+0645) Arabic Letter MEEM</td>
</tr>
<tr>
<td>NOON</td>
<td>0xBD</td>
<td>(U+0646) Arabic Letter NOON</td>
</tr>
<tr>
<td>HEH</td>
<td>0xBF</td>
<td>(U+0647) Arabic Letter HEH</td>
</tr>
</tbody>
</table>
The ANORM420 function transcodes the normalized string to the session encoding unless the modifier `t` is specified.

The ANORM420 function remaps the zero-length-break-space character (U+200B), also abbreviated ZWSP, to a space character unless the modifier `z` is specified. The zero-length character is located at code position 0x45.

Example

```plaintext
data _null_;  
a = '59CD57BC577745'x  ;  
s1 = anorm420(a) ;  
/* Turn off addition of space and mapping of Arabic-Indic numbers */  
s2 = anorm420(a,"si") ;  
/* Turn off transcoding */  
s3 = anorm420(a,'t') ;  
   put s1= $hex20. / s2= $hex20. / s3=$hex20. ;  
run;
```

These statements produce these results:

```
s1=D8A8D987D8A7D985D8A7
s2=D8A8D987D8A7D985D8A7
s3=58BF56BB567740404040
```

CALL KSCANX Routine

Returns the position and length of the `n`th word from a character string.

Categories: CAS  
             DBCS

Notes: CALL KSCANX is designed to process multi-byte data. However, if the first argument, `string`, has single-byte characters, then the CALL KSCANX routine processes the SBCS data. For more information, see “Internationalization Compatibility for SAS String Functions” on page 343.
Argument types for arguments that are updated must match in CALL routines. All argument types must be CHAR, VARCHAR, or NUMERIC. If the argument types do not match, a warning is issued to the SAS log.

Syntax

CALL KSCANX(<string>, count, position, length,<character-list>,<modifier(s)>);

Required Arguments

string
specifies a character constant, variable, or expression.

count
specifies a nonzero numeric constant, variable, or expression that has an integer value. The integer value specifies the number of the word in the character string that you want CALL KSCAN to select. For example, a value of 1 indicates the first word, a value of 2 indicates the second word, and so on. The following rules apply:

- If count is positive, CALL KSCANX counts words from left to right in the character string.
- If count is negative, CALL KSCANX counts words from right to left in the character string.

position
specifies a numeric variable in which the position of the word is returned. If count exceeds the number of words in the string, then the value that is returned in position is 0. If count is 0 or missing, then the value that is returned in position is missing.

length
specifies a numeric variable in which the length of the word is returned. If count exceeds the number of words in the string, then the value that is returned in length is 0. If count is 0 or missing, then the value that is returned in length is missing.

Optional Arguments

character-list
specifies an optional character expression that initializes a list of characters. This list determines which characters are used as the delimiters that separate words. The following rules apply:

- By default, all characters in character-list are used as delimiters.
- If you specify the K modifier in the modifier argument, then all characters that are not in character-list are used as delimiters.

TIP You can add more characters to character-list by using other modifiers.

modifier(s)
specifies a character constant, variable, or expression in which each non-blank character modifies the actions of the CALL KSCANX. Blanks are ignored. Use the following characters as modifiers:
a or A  adds alphabetic characters to the list of characters.

b or B  scans backward, from right to left instead of from left to right, regardless of the sign of the count argument.
c or C  adds control characters to the list of characters.
d or D  adds digits to the list of characters.
f or F  adds an underscore and English letters (that is, valid first characters in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
g or G  adds graphic characters to the list of characters. Graphic characters are those that, when printed, produce an image on paper.
h or H  adds a horizontal tab to the list of characters.
i or I  ignores the case of the characters.
k or K  treats all characters that are not in the list of characters as delimiters. That is, if K is specified, then characters that are in the list of characters are kept in the returned value rather than being omitted because they are delimiters. If K is not specified, then all characters that are in the list of characters are treated as delimiters.
l or L  adds lowercase letters to the list of characters.
m or M  specifies that multiple consecutive delimiters and delimiters at the beginning or end of the string argument refer to words that have a length of 0. If the M modifier is not specified, then multiple consecutive delimiters are treated as one delimiter, and delimiters at the beginning or end of the string argument are ignored.
n or N  adds digits, an underscore, and English letters (that is, the characters that can appear in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
o or O  processes the character-list and modifier arguments only once, rather than every time the CALL KSCANX routine is called. Using the O modifier in the DATA step can make CALL KSCANX run faster when you call it in a loop where the character-list and modifier arguments do not change. The O modifier applies separately to each instance of the CALL KSCANX routine in your SAS code. It does not cause all instances of the CALL KSCANX routine to use the same delimiters and modifiers.
p or P  adds punctuation marks to the list of characters.
q or Q  ignores delimiters that are inside substrings that are enclosed in quotation marks. If the value of the string argument contains unmatched quotation marks, then scanning from left to right produces different words than scanning from right to left.
s or S  adds space characters to the list of characters (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed).
t or T  trims trailing blanks from the string and character-list arguments. If you want to remove trailing blanks from just one character argument instead of both character arguments, then use the TRIM function instead of the CALL KSCANX routine with the T modifier.
Details

Definition of Delimiter and Word

A delimiter is any of several characters that are used to separate words. You can specify the delimiters in the character-list and modifier arguments.

In the CALL KSCANX routine, a word refers to a substring that has all of the following characteristics:

- is bounded on the left by a delimiter or the beginning of the string.
- is bounded on the right by a delimiter or the end of the string.
- contains no delimiters.

A word can have a length of 0 if there are delimiters at the beginning or end of the string or if the string contains two or more consecutive delimiters. However, the CALL KSCANX routine ignores words that have a length of 0 unless you specify the M modifier.

Using Default Delimiters in ASCII and EBCDIC Environments

If you use the CALL KSCANX routine with only four arguments, then the default delimiters depend on whether your computer uses ASCII or EBCDIC characters.

- If your computer uses ASCII characters, then the default delimiters are: blank ! $ % & ( ) * + , - . / ; < ^
  - In ASCII environments that do not contain the ^ character, the CALL KSCANX routine uses the ~ character instead.
- If your computer uses EBCDIC characters, then the default delimiters are as follows: blank ! $ % & ( ) * + , - . / ; < ¬ | ¢

If you use the modifier argument without specifying any characters as delimiters, then the only delimiters used are those that are defined by the modifier argument. In this case, the lists of default delimiters for ASCII and EBCDIC environments are not used. In other words, modifiers add to the list of delimiters that are explicitly specified by the character-list argument. Modifiers do not add to the list of default modifiers.

Using the CALL KSCANX Routine with the M Modifier

If you specify the M modifier, then the number of words in a string is defined as one plus the number of delimiters in the string. However, if you specify the Q modifier, delimiters that are inside quotation marks are ignored.
If you specify the M modifier, the CALL KSCANX routine returns a positive position and a length of 0 if one of the following conditions is true:

- The string begins with a delimiter and you request the first word.
- The string ends with a delimiter and you request the last word.
- The string contains two consecutive delimiters and you request the word that is between the two delimiters.

If you specify a count that is greater in absolute value than the number of words in the string, then the CALL KSCANX routine returns a position and length of 0.

Using the CALL KSCANX Routine without the M Modifier

If you do not specify the M modifier, then the number of words in a string is defined as the number of maximal substrings of consecutive non-delimiters. However, if you specify the Q modifier, delimiters that are inside quotation marks are ignored.

If you do not specify the M modifier, then the CALL KSCANX routine does the following:

- ignores delimiters at the beginning or end of the string.
- treats two or more consecutive delimiters as if they were a single delimiter.

If the string contains no characters other than delimiters or if you specify a count that is greater in absolute value than the number of words in the string, then the CALL KSCANX routine returns a position and length of 0.

Finding a Designated Word as a Character String

To find a designated word as a character string after calling the CALL KSCANX routine, use the SUBSTRN function with the string, position, and length arguments:

\[ k\text{substr}(\text{string}, \text{position}, \text{length}); \]

Using Null Arguments

The CALL KSCANX routine allows character arguments to be null. Null arguments are treated as character strings with a length of 0. Numeric arguments cannot be null.

Example

This example uses Chinese characters:
data _null_;
length x $20 y $20;
text1='これは漢字関数のテストです。';
x='nonblank';
i=1;
do until(x=' ');
   x=kscan(text1,i,'はの');
   put x=;
   i=i+1;
end;
end;
y='nonblank';
i=-1;
do until(y=' ');
   y=kscan(text1,i,'はの');
   put y=;
   i=i-1;
end;
run;

These statements produce these results:

<table>
<thead>
<tr>
<th>pos</th>
<th>len</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>3</td>
</tr>
</tbody>
</table>

### BASECHAR Function

Converts characters to base characters.

**Categories:** CAS

**Character**

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

### Syntax

```
STR=BASECHAR(<instr>,<Unicode type>)
```

**Required Arguments**

- **str**
  - data string that is converted.

- **instr**
  - input data string.

- **Unicode type**
  - If one of the following Unicode character formats is specified, national characters are represented in the specified format:

  - **ESC** Unicode Escape (\u00cd).
NCR  Numeric Character Representation (&#x00c5;).
PAREN Unicode Parenthesis Escape (&u00c5;).

Details
The BASECHAR function reads characters and converts them to base characters. Some characters consist of a base character and one or more accents or combining characters. The BASECHAR function reads the characters in the string and converts them to the base character form without the accents based on Unicode specifications. When Unicode type is specified, national characters are represented in the specified format. For example, the data string Mühlenfließ is converted to Muhlenfließ.

Example
The SAS session encoding for this example is wlatin1.

Note: The BASECHAR function uses Unicode character decomposition mappings, so the sharp s, ß, does not have a Unicode decomposition. http://www.unicode.org/reports/tr44/#Character_Decomposition_Mappings

data cities;
  length name$24 name_ascii$24;
  input name;
  name_ascii = basechar(name);
  put name_ascii=;
  datalines;
  Mühlenfließ
  LaUña
  ZielonaGóra
  run;

These statements produce these results:

<table>
<thead>
<tr>
<th>name_ascii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muhlenfließ</td>
</tr>
<tr>
<td>LaUna</td>
</tr>
<tr>
<td>ZielonaGora</td>
</tr>
</tbody>
</table>

The session encoding for this example is wlatin1 and the ESC option is specified.

data cities;
  length name$24 name_ascii$80;
  input name;
  name_ascii = basechar(name, "ESC");
  put name_ascii=;
  datalines;
  Mühlenfließ
  LaUña
  ZielonaGóra
  run;

These statements produce these results:
The session encoding for this example is wlatin1 and the NCR option is specified.

data cities;
  length name$24 name_ascii$80;
  input name;
  name_ascii = basechar(name, "NCR");
  put name_ascii=;
  datalines;
  Mühlenfließ
  LaUña
  ZielonaGóra
run;

These statements produce these results:

ENCODCOMPAT Function

Verifies the transcoding compatibility between two encodings.

Category: Encoding

Syntax

ENCODCOMPAT(source1,<source2>)

Required Arguments

source1
  a character string that represents an encoding.

source2
  a character string that represents an encoding. This argument is optional.

Details

If you specify one encoding, the function verifies the compatibility of the specified encoding with the current SAS session encoding.

If you specify two encodings, the function verifies the compatibility of the two encodings.

The function compares two encoding identifiers and determines whether the data needs to be transcoded. Source1 is the source encoding. Source2 is the destination encoding. Transcoding 7-bit ASCII to another type of ASCII is compatible, but transcoding ASCII to 7-bit ASCII might not be compatible.
The ENCODCOMPAT function specifies the following values:

-1  Source1 is not a valid encoding name.
-2  Source2 is not a valid encoding name.
0   The encodings are not compatible. Transcoding is needed.
1   The encodings are compatible. Transcoding is not needed.
2   A newline character is detected.

Example

The session encoding for these examples is UTF-8.

```plaintext
data encoding;
  isCompat= EncodCompat('xyz');
  put isCompat=;
run;
```

These statements produce this result:

```
isCompat=-1
```

```plaintext
data encoding;
  isCompat= EncodCompat ('ascii');
  put isCompat=;
run;
```

These statements produce this result:

```
isCompat=1
```

```plaintext
data encoding;
  isCompat= EncodCompat("ebcdic1149","open_ed-1149");
  put isCompat=;
run;
```

These statements produce this result:

```
isCompat=2
```

```plaintext
data encoding;
  isCompat= EncodCompat ('cp1251','ebcdic1149');
  put isCompat=;
run;
```

These statements produce this result:

```
isCompat=0;
```

This code compares the Windows encoding cp1251, also known as WLATIN2, with open_ed-1149, Iceland OpenEdition EBCDIC.
ENCODISVALID Function

Verifies a valid encoding name.

Category: Encoding

Syntax

ENCODISVALID(source)

Required Argument

source

a character string that represents an encoding name.

Details

The ENCODISVALID function returns the following values:

0  the character string is not a valid encoding name.
1  the character string is a valid short encoding name.
2  the character string is a valid long encoding name.
3  the character string is a valid alias encoding name.

Example

data valid;
  isValid1=EncodIsValid('xyz');
  isValid2=EncodIsValid('wlt2');
  isValid3=EncodIsValid('wlatin2');
  isValid4=EncodIsValid('cp1250');
  put isValid1=;
  put isValid2=;
  put isValid3=;
  put isValid4;
run;

These statements produce these results:
GETLOCENV Function

Returns the current locale/language environment.

Category: Locale

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

GETLOCENV()

Details

The GETLOCENV function returns the locale/language environment value for a valid SAS locale. The following environment values are possible:

SBCS
   The SAS session encoding is SBCS (Single-Byte Character Set). SASWZSD is loaded for string manipulation.

DBCS
   The SAS session encoding is DBCS (Double-Byte Character Set). SASWZSD is loaded for string manipulation.

MBCS
   The SAS session encoding is Unicode(UTF8). SASWZSU is loaded for string manipulation.

If you receive a blank value, then the WZSS subsystem is not available. This action suggests a configuration or installation error.

Example

Data test;
   option locale=french_france;
   environ=getlocenv();
   put environ=;
run;

These statements produce this result:

environ=MBCS
GETPXLANGUAGE Function

Returns the current two-letter language code.

**Category:** Locale

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see *Internationalization Compatibility* on page 343.

**Syntax**

```
GETPXLANGUAGE()
```

**Details**

The GETPXLANGUAGE function returns the two-letter language code based on the current value of the `LOCALE=` SAS system option. The length of the language name is two characters. If the size of the variable that receives the value is less than two characters, the value is truncated.

**Example**

In the first example, the `LOCALE=` system option is set to French_France. The second example is set to German. The third example is set to English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france;</td>
<td></td>
</tr>
<tr>
<td>lang=getpxLanguage();</td>
<td>fr</td>
</tr>
<tr>
<td>put lang;</td>
<td></td>
</tr>
<tr>
<td>option locale=German;</td>
<td>de</td>
</tr>
<tr>
<td>lang=getpxLanguage();</td>
<td></td>
</tr>
<tr>
<td>put lang;</td>
<td></td>
</tr>
<tr>
<td>option locale=en_US;</td>
<td>en</td>
</tr>
<tr>
<td>lang=getpxLanguage();</td>
<td></td>
</tr>
<tr>
<td>put lang;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**System Options:**
GETPXLOCALE Function

Returns the POSIX locale value for a SAS locale.

Syntax

GETPXLOCALE(<source>)

Required Argument

<source>

is an optional argument that specifies a locale name.

Details

The GETPXLOCALE function returns the POSIX locale value for a valid SAS locale name. If you specify an invalid locale name, then a null string is returned. If you do not specify a value for the <source> argument, then the function returns the POSIX name for the current SAS session. The length of the POSIX locale name is five characters. If the size of the variable that receives the value is less than five characters, the value is truncated.

Example

In the first example, the LOCALE= system option is set to French_France. In the second example, the <source> argument is set to German_Germany. In the third example, the <source> argument is set to English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france;</td>
<td>fr_FR</td>
</tr>
<tr>
<td>locale=getpxLocale();</td>
<td></td>
</tr>
<tr>
<td>put locale;</td>
<td></td>
</tr>
</tbody>
</table>
locale=getpxLocale("german_germany");  de_DE
put locale;

locale=getpxLocale("english_unitedstates");  en_US
put locale;

See Also

System Options:
- “LOCALE System Option” on page 698

Functions:
- “GETPXLANGUAGE Function” on page 398
- “GETPXREGION Function” on page 400

GETPXREGION Function

Returns the current two-letter region code.

Category: Locale

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

GETPXREGION()

Details

The GETPXREGION function returns the two-letter region code based on the current LOCALE= SAS system option. The length of the region name is two characters. If the size of the variable that receives the value is less than two characters, the value is truncated.

Example

In this example, the LOCALE system option is set to French_France.

data test;
option locale=french_france;
  region=getpxRegion();
  put region;
run;
These statements produce this result:

```
FR
```

In this example, the LOCALE system option is set to German.
```
data test;
  option locale=german;
  region=getpxRegion();
  put region;
run;
```

These statements produce this result:

```
DE
```

In this example, the LOCALE system option is set to English_United States.
```
data test;
  option locale=en_US;
  region=getpxRegion();
  put region;
run;
```

These statements produce this result:

```
US
```

See Also

**System Options:**
- “LOCALE System Option” on page 698

**Functions:**
- “GETPXLOCALE Function” on page 399
- “GETPXLANGUAGE Function” on page 398

---

### KCHARLIST Function

Returns the character list according to the modifier.

**Categories:**
- CAS
- DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internalization Compatibility on page 343.
Syntax

\[ \text{str} = \text{KCHARLIST}(\text{modifier}) \]

Required Argument

**modifier**

specifies a character constant, variable, or expression in which each non-blank character modifies the action of the KCHARLIST function. Use the following characters as modifiers:

- A or a returns alphabetic characters.
- C or c returns control characters.
- D or d returns digital characters.
- For f returns the first characters in SAS names.
- G or g returns graphic characters.
- L or l returns lowercase characters.
- P or p returns punctuation characters.
- S or s returns all space characters.
- U or u returns uppercase characters.
- X or x returns hexadecimal characters.

Details

KCHARLIST returns encoding-specific characters that can be used with other K functions. Using KCHARLIST to determine the delimiters helps to avoid the ambiguity of the delimiters in SAS string functions.

KCHARLIST can return DBCS space and punctuation character lists.

Example

This example uses KCHARLIST and KSCAN to process a string with a DBCS space and an SBCS space.

```
data _null_
  string = "北京 赛仕 软件";                      /* 1 */
  result1 = KSCAN(string, 2);                    /* 2 */
  put result1=;                                  /* 3 */
  result2 = KSCAN(string, 2, KCHARLIST('S'));    /* 4 */
  put result2=;                                  /* 5 */
run;
```

1. Assign a value with a DBCS space and an SBCS space to the variable `string`. The input string contain two spaces. The first space is a DBCS space, and the second space is an SBCS space.

2. KSCAN selects the second word, but KSCAN ignores DBCS spaces, so 软件 is selected.

3. The value 软件 is written to the output.

4. KCHARLIST processes the DBCS spaces and passes the space list to KSCAN as delimiters. The correct value 赛仕 is selected.
The value 赛仕 is written to the output.

result1=软件
result2=赛仕

See Also

Functions

- “SCAN Function” in SAS Functions and CALL Routines: Reference
- “KSCAN Function” on page 445

KCOMPARE Function

Returns the result of a comparison of character expressions.

Categories: CAS DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: Non-DBCS equivalent function is the “COMPARE Function” in SAS Functions and CALL Routines: Reference.

Syntax

KCOMPARE(source,<pos, count, > findstr,<modifiers>)

Required Arguments

source
specifies the character expression to be compared.

findstr
specifies the character expression to compare to source.

Optional Arguments

pos
specifies the starting position in source to begin the comparison. If pos is omitted, the entire source is compared. If pos is less than 0, source is assumed as extended DBCS data that does not contain any SO/SI characters.

modifiers

i or I ignores the case in source and findstr.

l or L removes leading blanks in source and findstr before comparing the values.

n or N removes quotation marks from any argument that is a literal name and ignores the case of source and findstr. A literal name is a
name token that is expressed as a string within quotation marks, followed by the uppercase or lowercase letter n. Literal names enable you to use special characters (including blanks), which are not allowed in SAS data sets or variable names. For KCOMPARE to recognize a string as a literal name, the first character must be a quotation mark.

: (colon) truncates the longer string in source or findstr to the length of the shorter string, whichever is greater. If : modifier is not specified, then use the longer string length, and the shorter one will be padded with blanks at the end of the string.

count specifies the number of bytes to compare. If count is omitted, all of source that follows pos is compared, except for any trailing blanks.

Details
The order in which the modifiers appear in the COMPARE function is relevant.

- LN first removes leading blanks from each string, and then it removes quotation marks from name literals.
- NL first removes quotation marks from name literals, and then it removes leading blanks from each string.

KCOMPARE returns values as follows:
- a negative value if source is less than findstr
- 0 if source is equal to findstr
- a positive value if source is greater than findstr

Example
The following example uses Japanese characters.

options pagesize=1 nodate ls=80 ps=60;

data test;
  rc1 = kcompare('漢字のテスト', '漢字');
  rc2 = kcompare('漢字のテスト'7,'テスト');
  rc3 = kcompare('漢字のテスト'1,6,'漢字');
  rc4 = kcompare('漢字のテスト','ABC');
  rc5 = kcompare('ABCDEF','ABC');
run;

proc print data=test; run;

The SAS System

Obs   rc1   rc2   rc3   rc4   rc5
     1     3     0     3     1     4
KCOMPRESS Function

Removes specified characters from a character expression.

Categories: CAS
            DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: Non-DBCS equivalent function is COMPRESS in SAS Functions and CALL Routines: Reference.

Syntax

KCOMPRESS(source,<characters-to-remove>, <modifier(s)>)

Required Argument

source specifies a character expression that contains the characters to be removed. When only source is specified, KCOMPRESS returns this expression with all of the single and double-byte blanks removed.

Optional Arguments

characters-to-remove specifies the character or characters that KCOMPRESS removes from the character. If characters-to-remove is omitted, KCOMPRESS removes all blank expressions.

Tip Enclose a literal string of characters in quotation marks.

modifier specifies a character constant, variable, or expression in which each non-blank character modifies the action of the KCOMPRESS function. Full-width alphabetic and numeric characters are supported. Blanks are ignored. The following characters can be used as modifiers:

a or A adds alphabetic characters to the list of characters. English characters and full-width alphabetic and numeric characters are supported.

c or C adds control characters to the list of characters.

d or D adds numeric characters (or digits) to the list of characters. English characters and full-width alphabetic and numeric characters are supported.

f or F adds the underscore character and English characters to the list of characters. Only English characters are supported.

g or G adds graphic characters to the list of characters.
h or H adds a horizontal tab to the list of characters.
i or I ignores the case of the characters and full-width characters to be kept or removed.
k or K keeps the characters in the list instead of removing them.
l or L adds lowercase characters to the list of characters. English characters and full-width alphabetic and numeric characters are supported.
n or N adds numeric characters, the underscore character, and English characters to the list of characters. Only English characters are supported.
o or O uses the O modifier in the DATA step (excluding WHERE clauses) or the SQL procedure to make KCOMPRESS run much faster when it is called in a loop, where the second and third arguments do not change.
p or P adds punctuation marks to the list of characters.
s or S removes all Unicode space characters in a Unicode SAS session.
t or T trims trailing blanks from the first and second arguments. Only English characters are supported.
u or U adds uppercase characters to the list of characters. English characters and full-width alphabetic and numeric characters are supported.
w or W adds printable characters to the list of characters.
x or X adds hexadecimal characters to the list of characters.

TIP If the modifier is a constant, enclose it in quotation marks. Specify multiple constants in a single set of quotation marks. Modifier can also be expressed as a variable or an expression.

Example
The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>result=kcompress('漢字のテスト','の');</td>
<td>result=漢字テスト</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:
- “KLEFT Function” on page 430
KCOUNT Function

Returns the number of double-byte characters in an expression.

Categories:
- CAS
- DBCS

Restriction:
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\[ \text{KCOUNT(source)} \]

Required Argument

\(\text{source}\)

- specifies the character expression to count.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

Example

The following examples use Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>text='abcあいう';</td>
<td></td>
</tr>
<tr>
<td>result=kcount(text);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>result=3</td>
</tr>
<tr>
<td>data casuser.encodedcompat;</td>
<td></td>
</tr>
<tr>
<td>text='abcわいう';</td>
<td></td>
</tr>
<tr>
<td>result=kcount(text);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>result=3</td>
</tr>
</tbody>
</table>
KCOUNTC Function

counts individual characters in a character string.

Categories: CAS DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8).

Syntax

\[ \text{KCOUNTC} \left( \text{string}, \text{character-list}, \langle \text{modifier(s)} \rangle \right) ; \]

Required Arguments

\text{string}

specifies a character constant, variable, or expression in which characters are counted.

\text{TIP} Enclose a literal string of characters in quotation marks.

\text{character-list}

specifies a character constant, variable, or expression that initializes a list of characters. The KCOUNTC function counts characters in this list, provided that you do not specify the V modifier in the modifier argument. If you specify the V modifier, then all characters that are not in this list are counted. You can add more characters to the list by using other modifiers.

\text{TIP} Enclose a literal string of characters in quotation marks.

If there are no characters in the list after processing the modifiers, the KCOUNTC function returns a 0.

Optional Argument

\text{modifier(s)}

specifies a character constant, variable, or expression in which each non-blank character modifies the actions of the KCOUNTC function. Blanks are ignored. The following characters, in uppercase or lowercase, can be used as modifiers:

- \text{blank} is ignored.
- \text{a or A} adds alphabetic characters to the list of characters.
- \text{b or B} scans \text{string} from right to left instead of from left to right.
- \text{c or C} adds control characters to the list of characters.
- \text{d or D} adds digits to the list of characters.
f or F adds an underscore and English letters (that is, valid first characters in a SAS variable name using VALIDVARNAME=V7) to the list of characters.

g or G adds graphic characters to the list of characters. Graphic characters are those that, when printed, produce an image on paper.

h or H adds a horizontal tab to the list of characters.

i or I ignores the case of the characters.

l or L adds lowercase letters to the list of characters.

n or N adds digits, an underscore, and English letters (that is, the characters that can appear in a SAS variable name using VALIDVARNAME=V7) to the list of characters.

o or O processes the character-list and modifier arguments only once, at the first call to this instance of the KCOUNTC function. If you change the value of character-list or modifier in subsequent calls, the change might be ignored by the KCOUNTC function.

p or P adds punctuation marks to the list of characters.

s or S adds space characters to the list of characters (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed).

t or T trims trailing blanks from the string and character-list arguments. If you want to remove trailing blanks from just one character argument instead of both character arguments, then use the TRIM function instead of the KCOUNTC routine with the T modifier.

u or U adds uppercase letters to the list of characters.

v or V counts characters that do not appear in the list of characters. If you do not specify this modifier, then the KCOUNTC function counts characters that do appear in the list of characters.

w or W adds printable (writable) characters to the list of characters.

x or X adds hexadecimal characters to the list of characters.

**TIP** If modifier is a constant, enclose it in quotation marks. Specify multiple constants in a single set of quotation marks.

Details

The Basics

The KCOUNTC function allows character arguments to be null. Null arguments are treated as character strings with a length of zero. If there are no characters in the list of characters to be counted, the KCOUNTC function returns a zero.

Comparisons

The KCOUNTC function counts individual characters in a character string, whereas the KCOUNTX function counts substrings of characters in a character string.
Example

This example uses Chinese characters:

```sas
data _null_;
  xyz='SAS是全球最大的软件公司之一，是全球商业智能和分析软件与服务领袖。'
  howmanythis=kcountw(xyz, '软件'); /* Count '款' and '件' occurrence */
  put howmanythis = ;
run;
```

```
howmanythis=4
```

KCOUNTW Function

Counts the number of words in a character string.

**Categories:**
CAS
DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and is designed for DBCS data. However, if the first argument, `string`, has single-byte characters, then the KCOUNTW function processes the SBCS data. For more information, see "Internationalization Compatibility for SAS String Functions" on page 343.

**Syntax**

```
KCOUNTW(<string>,<character-list>,<modifier(s)>);
```

**Optional Arguments**

- `string` Specifies a character constant, variable, or expression in which words are counted.
- `character-list` Specifies an optional character constant, variable, or expression that initializes a list of characters. The characters in this list are delimiters that separate words. However, if you specify the K modifier in the modifier argument, then all characters, including the characters in the list, are treated as delimiters. To add more characters to the list, use other modifiers. `Delimiter` refers to any of several characters that you can specify to separate words.

**TIP** Character arguments can be null. Null arguments are treated as character strings with a length of zero. Numeric arguments cannot be null.

- `modifier` Specifies a character constant, variable, or expression in which each non-blank character modifies the actions of the KCOUNTW function. The following characters can be used as modifiers:
blank is ignored.
a or A adds alphabetic characters to the list of characters.
b or B counts from right to left instead of from left to right. Right-to-left counting makes a difference only when you use the Q modifier and the string contains unbalanced quotation marks.
c or C adds control characters to the list of characters.
d or D adds digits to the list of characters.
f or F adds an underscore and English letters (that is, valid first characters in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
g or G adds graphic characters to the list of characters. Graphic characters are those that, when printed, produce an image on paper.
h or H adds a horizontal tab to the list of characters.
i or I ignores the case of the characters.
k or K treats all characters that are not in the list as delimiters. If K is not specified, all characters that are in the list are treated as delimiters.
l or L adds lowercase letters to the list of characters.
m or M specifies that multiple consecutive delimiters and delimiters at the beginning or end of the string argument refer to words that have a length of zero. If the M modifier is not specified, multiple consecutive delimiters are treated as one delimiter, and delimiters at the beginning or end of the string argument are ignored.
n or N adds digits, an underscore, and English letters (that is, the characters that can appear in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
o or O processes the character-list and modifier arguments only once rather than every time the KCOUNTW function is called. Using the O modifier in the DATA step (excluding WHERE clauses) or in the SQL procedure can make the KCOUNTW function run faster when you call it in a loop where chars and modifier arguments do not change.
p or P adds punctuation marks to the list of characters.
q or Q ignores delimiters that are inside substrings that are enclosed in quotation marks. If the value of string contains unmatched quotation marks, scanning from left to right produces different words than scanning from right to left.
s or S adds space characters to the list of characters (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed).
t or T trims trailing blanks from the string and character-list arguments.
u or U adds uppercase letters to the list of characters.
w or W adds printable (writable) characters to the list of characters.
x or X adds hexadecimal characters to the list of characters.
Details

Definition of Word
In the KCOUNTW function, word refers to a substring that has one of these characteristics:
- is bounded on the left by a delimiter or the beginning of the string.
- is bounded on the right by a delimiter or the end of the string.
- contains no delimiters, except if you use the Q modifier and the delimiters are within substrings that have quotation marks.

Using the KCOUNTW Function in ASCII and EBCDIC Environments
If you use the KCOUNTW function with only two arguments, the default delimiters are different depending on whether your computer uses ASCII or EBCDIC characters.
- If your computer uses ASCII characters, here are the default delimiters: blank ! $ % & ( ) * + - . / ; < ^ | .
  In ASCII environments that do not contain the ^ character, the SCAN function uses the ~ character instead.
- If your computer uses EBCDIC characters, here are the default delimiters: blank ! $ % & ( ) * + - . / ; < ¬ | ¢

Using the M Modifier
If you do not use the M modifier, then a word must contain at least one character. If you use the M modifier, then a word can have a length of zero. In the latter case, the number of words is one plus the number of delimiters in the string, not the number of delimiters inside strings that are enclosed in quotation marks when you use the Q modifier.

Example
This example uses Chinese characters:

```plaintext
data null;
  xyz='SAS是全球最大的软件公司之一，是全球商业智能和分析软件与服务领袖。';
  howmanythis=kcountw(xyz, ' ', 'p'); /* Count words, use punctuations as delimiter. */
  put howmanythis =;
run;
```

KCOUNTX Function
Counts the number of times that a specified substring appears within a character string.
Categories: CAS  DBCS
This function is assigned an I18N Level 2 status and is designed for DBCS data. However, if the first argument, `string`, has single-byte characters, then the `KCOUNTX` function processes the SBCS data. For more information, see “Internationalization Compatibility for SAS String Functions” on page 343.

**Syntax**

```sas
KCOUNTX(string,substring,<modifier(s)>);
```

**Required Arguments**

**string**

specifies a character constant, variable, or expression in which substrings are counted.

**TIP** Enclose a literal string of characters in quotation marks.

**substring**

is a character constant, variable, or expression that specifies the substring of characters to count in `string`.

**TIP** Enclose a literal string of characters in quotation marks.

**Optional Argument**

**modifier**

is a character constant, variable, or expression that specifies one or more modifiers. The following **modifiers** can be in uppercase or lowercase:

- **i** ignores character case during the count. If this modifier is not specified, the `KCOUNTX` function counts only character substrings with the same case as the characters in `substring`.

- **t** trims trailing blanks from `string` and `substring`.

**TIP** If `modifier` is a constant, enclose it in quotation marks. Specify multiple constants in a single set of quotation marks. `Modifier` can also be expressed as a variable or as an expression.

**Details**

**Basics**

The `KCOUNTX` function searches `string`, from left to right, for the number of occurrences of the specified `substring` and returns that number of occurrences. If the substring is not found in `string`, the `KCOUNTX` function returns a value of 0.

**CAUTION**
If two occurrences of the specified substring overlap in the string, the result is undefined. For example, kcount('boobooboo', 'booboo') might return either a 1 or a 2.

Example

This example uses Chinese characters:

```plaintext
data _null_;
xyz='SAS是全球最大的软件公司之一，是全球商业智能和分析软件与服务领袖。';
howmanythis=kcount(xyz, '软件'); /* Count '软件' occurrence */
put howmanythis =;
run;
```

```
howmanythis=2
```

KCVT Function

Converts data from one type of encoding data to another type of encoding data.

**Categories:**
CAS
Character

**Restriction:**
This function is assigned an I18N Level 1 status. If possible, avoid I18N Level 1 functions if you are using a non-English language. Under certain circumstances, the I18N Level 1 functions might not work correctly with Double-Byte Character Set (DBCS) or Multi-Byte Character Set (MBCS) encodings. For more information, see Internationalization Compatibility on page 343.

**Syntax**

```
KCVT(text, intype, outtype, <options,...>)
```

**Required Arguments**

- `text`
  specifies the character variable to be converted.

- `intype`
  specifies the encoding of the data. The encoding of the text must match the input data's encoding. For valid values, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

  ASCIIANY and EBCDICANY are invalid encoding values.

- `outtype`
  specifies the encoding to be converted into character data. For valid values, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

  ASCIIANY and EBCDICANY are invalid encoding values.
**options**

specifies character data options. Here are the available options:

- **NOSOSI | NOSHIFT**
  No shift code or Hankaku characters.

- **INPLACE**
  Replaces character data by conversion. The INPLACE option is specified to secure the same location between different hosts whose lengths of character data are not identical. For example, the INPLACE option converts data from the host that requires Shift-Codes into the other host, which does not require Shift Codes. Truncation occurs when the length of the character data that is converted into `outtype` for Shift-Codes is longer than the length that is specified in `intype`.

- **KANA**
  Includes Hankaku katakana characters in columns of character data.

- **UPCASE**
  Converts a 2-byte alphabet to uppercase characters.

- **LOWCASE**
  Converts a 2-byte alphabet to lowercase characters.

- **KATA2HIRA**
  Converts katakana data to hiragana.

- **HIRA2KATA**
  Converts hiragana data to katakana.

**Details**

See "Internationalization Compatibility for SAS String Functions" on page 343 for restrictions and more information.

The KCVT function converts SBCS, DBCS, and MBCS character strings into encoding data. For example, the KCVT function can convert the following: ASCII code data to UCS2 encoding data, Greek code data to UTF-8, and Japanese SJIS code data to another Japanese code data. You can specify the following types for the `intype` and `outtype` options: UCS2, UCS2L, UCS2B, and UTF8. To enable the DBCS mode, specify the DBCS option in the configuration file or from the command line.

If the KCVT function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

In the SQL procedure, or in a WHERE clause in any procedure, the maximum length of a word that is returned by the KCVT function is 200 characters.

**Example**

The following code converts IBM PC codes into DEC codes for the external text file specified as `my-input-file` and writes the results in OUTDD.

```sas
data _null_
  infile 'my-input-file'
  file outdd noprint
  input @1 text $char80.;
  text = kcvt(text, 'pcibm', 'dec');
  put @1 text $char80.;
run;
```
See Also

System options:
- “DBCS System Option: UNIX, Windows, and z/OS” on page 687

Procedure:
- Chapter 18, “DBCSTAB Procedure,” on page 761

KFIN Function

Searches for a specific substring of characters within a character string.

Categories:
- CAS
- DBCS

Restriction:
This function is assigned an I18N Level 2 status and is designed for DBCS data. However, if the first argument, string, has multi-byte characters, then the KFIN function processes the multi-byte data. For more information, see “Internationalization Compatibility for SAS String Functions” on page 343.

Syntax

\[ \text{KFIN}(\text{string, substring}, <\text{start-position}>, <\text{modifier(s)}>), \text{KFIN}(\text{string, substring}, <\text{modifier(s)}>), <\text{start-position}>) \]

Required Arguments

\textit{string} specifies a character constant, variable, or expression that is searched for substrings.

\textbf{TIP} Enclose a literal string of characters in quotation marks.

\textit{substring} is a character constant, variable, or expression that specifies the substring of characters to search for in \textit{string}.

\textbf{TIP} Enclose a literal string of characters in quotation marks.

Optional Arguments

\textit{modifier(s)} is a character constant, variable, or expression that specifies one or more modifiers. The following modifiers are valid:
i or I
ignores character case during the search. If this modifier is not specified, KFIND searches only for character substrings with the same case as the characters in substring.

t or T
trims trailing blanks from string and substring.

Note: If you want to remove trailing blanks from only one character argument instead of both (or all) character arguments, use the KTRIM function instead of the KFIND function with the T modifier.

**TIP**  If *modifier* is a constant, enclose it in quotation marks. Specify multiple constants in a single set of quotation marks. *Modifier* can also be expressed as a variable or as an expression.

**start-position**
is a numeric constant, variable, or expression with an integer value that specifies the position at which the search should start and the direction of the search.

Details
Basics
The KFIND function searches *string* for the first occurrence of the specified substring, and returns the position of that substring. If the substring is not found in string, the KFIND function returns a value of 0.

If *start-position* is not specified, the KFIND function starts the search at the beginning of the string and searches the string from left to right. If *start-position* is specified, the absolute value of *start-position* determines the position at which to start the search. The sign of *start-position* determines the direction of the search.

<table>
<thead>
<tr>
<th>Value of startpos</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than 0</td>
<td>Starts the search at position <em>start-position</em> and searches to the right. If <em>start-position</em> is greater than the length of <em>string</em>, KFIND returns a value of 0.</td>
</tr>
<tr>
<td>Less than 0</td>
<td>Starts the search at position <em>start-position</em> and searches to the left. If <em>start-position</em> is greater than the length of <em>string</em>, the search starts at the end of <em>string</em>.</td>
</tr>
<tr>
<td>Equal to 0</td>
<td>Returns a value of 0.</td>
</tr>
</tbody>
</table>

Example
This example uses Chinese characters:
**KFINDC Function**

Searches a string for any character in a list of characters.

**Categories:**
- CAS
- DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and is designed for DBCS data. However, if the first argument, `string`, has single-byte characters, then the KFINDC function processes the SBCS data. For more information, see "Internationalization Compatibility for SAS String Functions" on page 343.

**Note:**
This function supports the varchar type.

**Syntax**

\[
\text{KFINDC}(string,<\text{character-list}>)
\]

\[
\text{KFINDC}(string,\text{character-list},<\text{modifier(s)}>)
\]

\[
\text{KFINDC}(string,\text{character-list},<\text{modifier(s)},<\text{start-position}>)
\]

\[
\text{KFINDC}(string,\text{character-list},<\text{modifier(s)},<\text{start-position}>,<\text{modifier(s)}>)
\]

**Required Arguments**

`string`

is a character constant, variable, or expression that specifies the character string to search.

**TIP** Enclose a literal string of characters in quotation marks.

`character-list`

is a constant, variable, or character expression that initializes a list of characters. The KFINDC function searches for the characters in this list provided that you do not specify the K modifier in the modifier argument. If you specify the K modifier, the KFINDC function searches for all characters that are not in this list of characters. You can add more characters to the list by using other modifiers.

`modifier(s)`

is a character constant, variable, or expression in which each character modifies the actions of the KFINDC function. The following characters, in uppercase or lowercase, can be used as modifiers:

- `blank` is ignored.

---

```sas
data _null_
  xyz='SAS是全球最大的软件公司之一, 是全球商业智能和分析软件与服务领域。';
  pos_of=kfind(xyz, '软件'); /* Find first position of '软件' */
  put pos_of = 
run;

pos_of=10
```

---
a or A adds alphabetic characters to the list of characters.
b or B scans string from right to left instead of from left to right.
c or C adds control characters to the list of characters.
d or D adds digits to the list of characters.
f or F adds an underscore and English letters (that is, valid first characters in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
g or G adds graphic characters to the list of characters. Graphic characters are those that, when printed, produce an image on paper.
h or H adds a horizontal tab to the list of characters.
i or I ignores the case of the characters.
k or K searches for any character that does not appear in the list of characters. If you do not specify this modifier, then the KFINDC function searches for any character that appears in the list of characters. This modifier has the same functionality as the v or V modifier.
l or L adds lowercase letters to the list of characters.
n or N adds digits, an underscore, and English letters (that is, the characters that can appear in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
o or O processes the character-list and the modifier arguments only once rather than every time the KFINDC function is called. Using the O modifier in the DATA step (excluding WHERE clauses) or in the SQL procedure can make the KFINDC function run faster when you call it in a loop where the character-list and the modifier arguments do not change.
p or P adds punctuation marks to the list of characters.
s or S adds space characters to the list of characters (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed).
t or T trims trailing blanks from the string and character-list arguments. If you want to remove trailing blanks from just one character argument instead of both character arguments, then use the TRIM function instead of the KFINDC routine with the T modifier.
u or U adds uppercase letters to the list of characters.
v or V searches for any character that does not appear in the list of characters. If you do not specify this modifier, then KFINDC searches for any character that appears in the list of characters. This modifier has the same functionality as the k or K modifier.
w or W adds printable (writable) characters to the list of characters.
x or X adds hexadecimal characters to the list of characters.

TIP If modifier is a constant, then enclose it in quotation marks. Specify multiple constants in a single set of quotation marks. Modifier can also be expressed as a variable or as an expression.
Optional Argument

*start-position*

is an optional numeric constant, variable, or expression with an integer value that specifies the position at which the search should start and the direction in which to search.

Details

Basics

The KFINDC function searches string for the first occurrence of the specified characters and returns the position of the first character found. If no characters are found in string, then the KFINDC function returns a value of 0.

The KFINDC function allows character arguments to be null. Null arguments are treated as character strings that have a length of zero. Numeric arguments cannot be null.

If *start-position* is not specified, the KFINDC function begins the search at the end of the string if you use the B modifier or at the beginning of the string if you do not use the B modifier.

If *start-position* is specified, the absolute value of *start-position* specifies the position at which to begin the search. If you use the B modifier, the search always proceeds from right to left. If you do not use the B modifier, the sign of *start-position* specifies the direction in which to search. The following table summarizes the search directions:

<table>
<thead>
<tr>
<th>Value of <em>startpos</em></th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than 0</td>
<td>Search begins at position <em>start-position</em> and proceeds to the right. If <em>start-position</em> is greater than the length of the string, the KFINDC function returns a value of 0.</td>
</tr>
<tr>
<td>Less than 0</td>
<td>Search begins at position <em>start-position</em> and proceeds to the left. If <em>start-position</em> is less than the negative of the length of the string, the search begins at the end of the string.</td>
</tr>
<tr>
<td>Equal to 0</td>
<td>Returns a value of 0.</td>
</tr>
</tbody>
</table>

Example

This example uses Chinese characters.

data _null_;  
xyz='SAS是全球最大的软件公司之一，是全球商业智能和分析软件与服务领袖。';  
pos_of=kfindc(xyz, '软件的');  /* Find minimum position of '软'，'件' or'的' */  
put pos_of = ;  
run;
KFINDW Function

Returns the character position of a word in a string or the number of the word in a string.

Categories: CAS  
            DBCS  

Restriction: This function is assigned an I18N Level 2 status and is designed for DBCS data. However, if the first argument, string, has single-byte characters, then the KFINDW function processes the SBCS data. For more information, see “Internationalization Compatibility for SAS String Functions” on page 343.

Syntax

\[
\text{KFINDW}(\text{string}, \text{word}, \text{character-list}) \\
\text{KFINDW}(\text{string}, \text{word}, \text{character-list}, \text{modifier(s)}, \text{<start-position>}) \\
\text{KFINDW}(\text{string}, \text{word}, \text{character-list}, \text{start-position}, \text{<modifier(s)>}) \\
\text{KFINDW}(\text{string}, \text{word}, \text{start-position}, \text{<character-list}}, \text{modifier(s)})
\]

Required Arguments

**string**  
is a character constant, variable, or expression that specifies which character string to search.

**word**  
is a character constant, variable, or expression that specifies the word to search for in string.

**character-list**  
is an optional character constant, variable, or expression that initializes a list of characters.

The characters in this list are the delimiters that separate words, provided that you do not specify the K modifier in the modifier argument. If you specify the K modifier, then all characters that are not in this list are delimiters. You can add more characters to this list by using other modifiers.

**start-position**  
is an optional numeric constant, variable, or expression with an integer value that specifies the position at which the search should begin and the direction in which to search.

**modifier**  
specifies a character constant, variable, or expression in which each non-blank character modifies the actions of the KFINDW function.

**TIP** If you use the modifier argument, then it must be positioned after the character-list argument.
You can use these characters as modifiers:

- blank is ignored.
- a or A adds alphabetic characters to the list of characters.
- b or B scans *string* from right to left instead of from left to right.
- c or C adds control characters to the list of characters.
- d or D adds digits to the list of characters.
- e or E counts the words that are scanned until the specified word is found instead of determining the character position of the specified word in the string. Fragments of a word are not counted.
- f or F adds an underscore and English letters (that is, valid first characters in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
- g or G adds graphic characters to the list of characters. Graphic characters are those that, when printed, produce an image on paper.
- h or H adds a horizontal tab to the list of characters.
- i or I ignores the case of the characters.
- k or K treats all characters that are not in the list of characters as delimiters. If K is not specified, then all characters that are in the list of characters are treated as delimiters.
- l or L adds lowercase letters to the list of characters.
- n or N adds digits, an underscore, and English letters (that is, the characters that can appear in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
- o or O processes the *character-list* and *modifier* arguments only once rather than every time the KFINDW function is called. Using the O modifier in the DATA step (excluding WHERE clauses) or in the SQL procedure can make the KFINDW function run faster when you call it in a loop where the *character-list* and *modifier* arguments do not change.
- p or P adds punctuation marks to the list of characters.
- q or Q ignores delimiters that are inside substrings that are enclosed in quotation marks. If the value of the string argument contains unmatched quotation marks, then scanning from left to right produces different words than scanning from right to left.
- r or R removes leading and trailing delimiters from the word argument.
- s or S adds space characters to the list of characters (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed).
- t or T trims trailing blanks from the *string*, *word*, and *character-list* arguments.
- u or U adds uppercase letters to the list of characters.
- w or W adds printable (writable) characters to the list of characters.
- x or X adds hexadecimal characters to the list of characters.
Details

Delimiter

*Delimiter* refers to any of several characters that are used to separate words. You can specify the delimiters by using the *character* argument, the *modifier* argument, or both. If you specify the Q modifier, then the characters inside substrings that are enclosed in quotation marks are not treated as delimiters.

Word

*Word* refers to a substring that has both of the following characteristics:

- bounded on the left by a delimiter or the beginning of the string
- bounded on the right by a delimiter or the end of the string.

*Note:* A word can contain delimiters. In this case, the KFINDW function differs from the SCAN function, in which words are defined as containing no delimiters.

Searching for a String

If the KFINDW function fails to find a substring that both matches the specified word and satisfies the definition of a word, then the KFINDW function returns a value of 0.

If the KFINDW function finds a substring that both matches the specified word and satisfies the definition of a word, the value that is returned by the KFINDW function depends on whether the E modifier is specified:

- If you specify the E modifier, then the KFINDW function returns the number of complete words that were scanned while searching for the specified word. If *start-position* specifies a position in the middle of a word, then that word is not counted.
- If you do not specify the E modifier, then the KFINDW function returns the character position of the substring that is found.

If you specify the *start-position* argument, then the absolute value of *start-position* specifies the position at which to begin the search. The sign of *start-position* specifies the direction in which to search:

<table>
<thead>
<tr>
<th>Value of <em>startpos</em></th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than 0</td>
<td>Search begins at <em>start-position</em> and proceeds to the right. If <em>start-position</em> is greater than the length of the string, then the KFINDW function returns a value of 0.</td>
</tr>
<tr>
<td>Less than 0</td>
<td>Search begins at position <em>start-position</em> and proceeds to the left. If <em>start-position</em> is less than the negative of the length of the string, then the search begins at the end of the string.</td>
</tr>
<tr>
<td>Equal to 0</td>
<td>The KFINDW function returns a value of 0.</td>
</tr>
</tbody>
</table>
If you do not specify the start-position argument or the B modifier, then the KFINDW function searches from left to right starting at the beginning of the string. If you specify the B modifier, but do not use the start-position argument, then the KFINDW function searches from right to left starting at the end of the string.

Using the KFINDW Function in ASCII and EBCDIC Environments

If you use the KFINDW function with only two arguments, the default delimiters depend on whether your computer uses ASCII or EBCDIC characters.

- If your computer uses ASCII characters, then the default delimiters are as follows: blank ! $ % & ( ) * + , - . / ; < ^ |
  In ASCII environments that do not contain the ^ character, the KFINDW function uses the ~ character instead.
- If your computer uses EBCDIC characters, then the default delimiters are as follows: blank ! $ % & ( ) * + , - . / ; < ¬ | ¢

Using Null Arguments

The KFINDW function allows character arguments to be null. Null arguments are treated as character strings with a length of zero. Numeric arguments cannot be null.

Processing SBCS and DBCS Data

The KFINDW function is designed to process SBCS data, but it can process DBCS data with certain conditions. Here are the criteria for SBCS and DBCS processing:

- If string is declared as varchar and you are processing multi-byte data, then the KFINDW function processes DBCS.
- If string is not declared as varchar or you are processing single-byte data, then the KFINDW function processes SBCS.

Examples:

Example 1

This example uses Chinese characters:

```cpp
data _null_;
xyz='SAS是全球最大的 軟件 公司之一，SAS是全球商业智能和分析软件与服务领袖。';
pos_of=kfindw(xyz, ' 軟件'); /* The default delimiter is SBCS blank. */
put pos_of = ;
run;
pos_of=11
```

Example 2

Here is another example:
data _null_;
xyz='SAS是全球最大的软件公司之一，sas是全球商业智能和分析软件与服务的领导者。';
pos_of=kfindx(xyz, 'sas', '', 'sp'); /* Use spaces and punctuations as delimiter. */
put pos_of = ;
run;

pos_of=17

KINDEX Function

Searches a character expression for a string of characters.

Categories: CAS DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: Non-DBCS equivalent function is INDEX in SAS Functions and CALL Routines: Reference

Syntax

KINDEX(source, excerpt)

Required Arguments

source
  specifies the character expression to search.

excerpt
  specifies the string of characters to search for in the character expression.

Tip  Enclose a literal string of characters in quotation marks.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The KINDEX function searches source, from left to right, for the first occurrence of the string that is specified in excerpt, and returns the position in source of the string's first character. If the string is not found in source, KINDEX returns a value of 0. If there are multiple occurrences of the string, KINDEX returns only the position of the first occurrence.

Example

The following example uses Japanese characters.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| data _null_;  
  text='漢字文字列の検索';  
  result=kindex(text,'検索');  
  put result=;  
  text2='探す';  
  result=kindex(text,text2);  
  put result=;  
  run;                | result=7  
  result=0 |

See Also

**Functions:**
- "KINDEXC Function" on page 428

**KINDEXB Function**

Searches a character expression for specified characters and returns byte-based values.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

**Syntax**

```
KINDEXB(source, excerpt)
```

**Required Arguments**

- `source` specifies the character expression to search for.
- `excerpt` specifies the string of characters to search for in the character expression.

**TIP** Enclose a literal string of characters in quotation marks.

**Details**

The KINDEXB function searches `source`, from left to right, for the first occurrence of the string that is specified in `excerpt` and returns the string's first character position in `source`. If the string is not found in `source`, KINDEXB returns a value of 0. If there
are multiple occurrences of the string, KINDEXB returns only the position of the first occurrence.

Comparisons

KINDEXB returns byte-based values. KINDEX returns character-based values. When processing an SBCS string (for example, wlatin1 encoding), KINDEXB and KINDEX return identical results. But in a DBCS session (for example, EUC-JP or SHIFT-JIS encoding), most CJK characters are 2 bytes wide, causing these functions to return different results.

The KINDEXCB function searches for the first occurrence of any individual character that is present within the character string. The KINDEXB function searches for the first occurrence of the character string as a pattern.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>text='漢字文字列の検索';</td>
<td></td>
</tr>
<tr>
<td>result=kindexb(text, '検索');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>text2='探す';</td>
<td></td>
</tr>
<tr>
<td>result=kindexb(text,text2);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result=13</td>
</tr>
<tr>
<td></td>
<td>Result=0</td>
</tr>
</tbody>
</table>

KINDEXCB Function

Searches a character expression for specified characters and returns byte-based values.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\[
\text{KINDEXCB(} \text{source, excerpt-1, ..., excerpt-n)}
\]

Required Arguments

\textit{source} specifies the character expression to search for.

\textit{excerpt-1, ..., excerpt-n} specifies the characters to search for in the character expression.
Details

The KINDEXCB function searches the data in source, from left to right, for the first occurrence of any character in the excerpts’ arguments and returns the position in source of that character. If none of the characters in excerpt-1 through excerpt-n in source are found, KINDEXCB returns a value of 0.

Comparisons

KINDEXCB returns byte-based values. KINDEXC returns character-based values. When processing SBCS strings (for example, wlatin1 encoding), both functions return identical results. But in a DBCS session (for example, EUC-JP or SHIFT-JIS encoding), most CJK characters are 2 bytes wide, causing these functions to return different results.

The KINDEXCB function searches for the first occurrence of any individual character that is present within the character string, whereas the KINDEXB function searches for the first occurrence of the character string as a pattern.

Example

The following example uses Japanese characters.

```
data _null_;  
a='ＡＢＣデフ（漢字123456）';  
result=kindexcb(a, '1234567890', '漢漢');  
put result=;  
run;  
```

KINDEXC Function

Searches a character expression for specified characters and returns character-based values.

Categories: CAS

DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: The Non-DBCS equivalent function is "INDEXC Function" in SAS Functions and CALL Routines: Reference
Syntax

KINDEXC(\textit{source, excerpt-1, ...excerpt-n})

Required Arguments

\textit{source}

specifies the character expression to search for.

\textit{excerpt-1, ...excerpt-n}

specifies the characters to search for in the character expression.

Tips

If you specify more than one excerpt, separate them with a comma.

Enclose a literal string of characters in quotation marks.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The KINDEXC function searches \textit{source}, from left to right, for the first occurrence of any character present in the excerpts and returns the position in \textit{source} of that character. If none of the characters in \textit{excerpt-1} through \textit{excerpt-n} in \textit{source} are found, KINDEXC returns a value of 0.

Comparisons

The KINDEXC function searches for the first occurrence of any individual character that is present within the character string, whereas the KINDEX function searches for the first occurrence of the character string as a pattern.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=9</td>
</tr>
<tr>
<td>a=’A B C. def (漢字123456)’;</td>
<td></td>
</tr>
<tr>
<td>result=kindexc(a,’1234567890’,’感漢’);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Function:

- “KINDEX Function” on page 425
KLEFT Function

Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.

Categories: CAS
DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Note: This function supports the VARCHAR data type.

Tip: Non-DBCS equivalent function is LEFT in SAS Functions and CALL Routines Reference.

Syntax

KLEFT(argument)

Required Argument

argument specifies any SAS character expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

KLEFT returns an argument and removes the leading blanks.

The KLEFT, KRIGHT, and KTRIM functions remove the following Unicode space characters.

Table 13.6 Unicode Spaces That Are Removed by KLEFT, KRIGHT, and KTRIM

<table>
<thead>
<tr>
<th>Unicode Spaces Definition in ANYSPACE</th>
<th>Unicode Spaces That Are Removed by KLEFT, KRIGHT, and KTRIM</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+0009</td>
<td>CHARACTER TABULATION</td>
<td></td>
</tr>
<tr>
<td>U+000A</td>
<td>LINE FEED</td>
<td></td>
</tr>
<tr>
<td>U+000B</td>
<td>LINE TABULATION</td>
<td></td>
</tr>
<tr>
<td>U+000C</td>
<td>FORM FEED</td>
<td></td>
</tr>
<tr>
<td>U+000D</td>
<td>CARRIAGE RETURN</td>
<td></td>
</tr>
<tr>
<td>Unicode Spaces Definition in ANYSPACE</td>
<td>Unicode Spaces That Are Removed by KLEFT, KRIGHT, and KTRIM</td>
<td>Comment</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>U+0020</td>
<td>U+0020</td>
<td>SINGLE-BYTE SPACE, DEPEND ON COMPILING</td>
</tr>
<tr>
<td></td>
<td>U+0085</td>
<td>NEXT LINE</td>
</tr>
<tr>
<td>U+00A0</td>
<td>U+00A0</td>
<td>NO-BREAK SPACE</td>
</tr>
<tr>
<td>U+1680</td>
<td>U+1680</td>
<td>OGHAM SPACE MARK</td>
</tr>
<tr>
<td>U+2000</td>
<td>U+2000</td>
<td>EN QUAD</td>
</tr>
<tr>
<td>U+2001</td>
<td>U+2001</td>
<td>EM QUAD</td>
</tr>
<tr>
<td>U+2002</td>
<td>U+2002</td>
<td>EN SPACE</td>
</tr>
<tr>
<td>U+2003</td>
<td>U+2003</td>
<td>EM SPACE</td>
</tr>
<tr>
<td>U+2004</td>
<td>U+2004</td>
<td>THREE-PER-EM SPACE</td>
</tr>
<tr>
<td>U+2005</td>
<td>U+2005</td>
<td>FOUR-PER-EM SPACE</td>
</tr>
<tr>
<td>U+2006</td>
<td>U+2006</td>
<td>SIX-PER-EM SPACE</td>
</tr>
<tr>
<td>U+2007</td>
<td>U+2007</td>
<td>FIGURE-EM SPACE</td>
</tr>
<tr>
<td>U+2008</td>
<td>U+2008</td>
<td>PUNCTUATION SPACE</td>
</tr>
<tr>
<td>U+2009</td>
<td>U+2009</td>
<td>THIN SPACE</td>
</tr>
<tr>
<td>U+200A</td>
<td>U+200A</td>
<td>HAIR SPACE</td>
</tr>
<tr>
<td>U+2028</td>
<td>U+2028</td>
<td>U+2028 LINE SEPARATOR</td>
</tr>
<tr>
<td>U+2029</td>
<td>U+2029</td>
<td>U+2029 PARAGRAPH SEPARATOR</td>
</tr>
<tr>
<td>U+202f</td>
<td>U+202F</td>
<td>NARROW NO-BREAK SPACE</td>
</tr>
<tr>
<td>U+205f</td>
<td>U+205F</td>
<td>MEDIUM MATHEMATICAL SPACE</td>
</tr>
<tr>
<td>U+3000</td>
<td>U+3000</td>
<td>IDEOGRAPHIC SPACE</td>
</tr>
</tbody>
</table>
Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data _null_;</code></td>
<td><code>-------1-------2-----+</code></td>
</tr>
<tr>
<td><code>a=’漢字のテスト’;</code></td>
<td>漢字のテスト</td>
</tr>
<tr>
<td><code>b=kleft(a);</code></td>
<td>漢字のテスト</td>
</tr>
<tr>
<td><code>put ’-------1-------2-----’;</code></td>
<td>漢字のテスト</td>
</tr>
<tr>
<td><code>put a $25.;</code></td>
<td></td>
</tr>
<tr>
<td><code>put b $25.;</code></td>
<td></td>
</tr>
<tr>
<td><code>run;</code></td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:
- “KCOMPRESS Function” on page 405
- “KRIGHT Function” on page 444
- “KTRIM Function” on page 460

KLENGTH Function

Returns the length of an argument.

Categories:
- CAS
- DBCS

Restriction:
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip:
Non-DBCS equivalent function is LENGTH in SAS Functions and CALL Routines: Reference.

Syntax

`KLENGTH(argument)`

Required Argument

`argument`
- specifies any SAS expression.
Details
See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The KLENGTH function returns an integer that represents the position of the rightmost non-blank character in the argument. If the value of the argument is missing, KLENGTH returns a value of 0. If the argument is an uninitialized numeric variable, KLENGTH returns a value of 12 and prints a note in the SAS log that the numeric values have been converted to character values.

If you are using SBCS EnglishVARCHAR data, performance issues might occur. To improve the performance, use the following code with non-K functions.

This code improves performance and the results reflect the KLENGTH function's behavior.

```
len = LENGTHC(TRIMN(x));
if NOT len then len = 1;
```

Example
The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=6</td>
</tr>
<tr>
<td>text='abcあいう';</td>
<td></td>
</tr>
<tr>
<td>result=klength(text);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KLOWCASE Function
Converts the uppercase alphabetic letters to lowercase letters.

Categories: CAS
           DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: Non-DBCS equivalent function is LOWCASE in SAS Functions and CALL Routines: Reference.

Syntax

```
KLOWCASE(argument)
```
Required Argument

**argument**
specifies any SAS character expression.

Details

The KLOWCASE function copies a character argument, converts the uppercase alphabetic letters to lowercase letters, and returns the altered value.

The definitions of the alphabetic letters differ based on the SAS session encoding. Here are typical SAS sessions and how KLOWCASE processes alphabetic letters for the different sessions.

**UTF-8 (SAS Unicode server)**

When the SAS session encoding is UTF-8, the Unicode character attributes apply. If a character is classified as a letter and has a lowercase equivalent, KLOWCASE returns the lowercase letter. Otherwise, the same letter is returned.

**DBCS (For example, Chinese, Japanese and Korean languages)**

KLOWCASE converts uppercase letters in the English alphabet to lowercase in a DBCS SAS session.

**SBCS (For example, English or European languages)**

When the SAS session encoding is SBCS, the TRANTAB option is updated to use the mapping tables for uppercase letters and the lowercase letter equivalents. The mapping tables for lowercase letters use the naming pattern of `nnnn_lcs`, where `nnnn` matches the short encoding name. The KLOWCASE function uses these tables to convert the uppercase letters to the lowercase letters that are supported by the session encoding.

Examples:

**Example 1**

In this example, the text contains a mix of English alphabetic uppercase letters plus the uppercase Ñ. The SAS session encoding is Windows code page 1252, which is WLATIN1. Notice that KLOWCASE returns all the letters in lowercase, including the Ñ. This example demonstrates results from KLOWCASE when the SAS encoding is SBCS and supports Western European characters. The same result is returned for UTF-8 text when the SAS encoding is UTF-8.

```sas
options locale=en_US;
data one;
x='NIÑO';
y=klowcase(x);
put x=;
p y=;
run;
```

```
x=NIÑO
y=niño
```

**Example 2**

This example uses a mix of Japanese and English alphabetic uppercase letters. In the result from KLOWCASE, the English alphabetic letters are converted to
lowercase and the Japanese characters are unchanged. The example demonstrates results from KLOWCASE when the SAS session uses a DBCS encoding that supports Japanese characters or UTF-8.

```sas
data _null_;
  result=klowcase('ABCあいうえお');
  put result=;
run;

result=abcあいうえお
```

---

**KPROPCASE Function**

Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see [Internationalization Compatibility](#) on page 343.

**Syntax**

```
str=KPROPCASE(<instr> , (<options> ))
```

**Required Arguments**

- `str` data string that has been converted and is in the current SAS session encoding.
- `instr` input data string.
- `options` converts Japanese, Chinese, Korean, and Taiwanese characters based on specified options.

**HALF-KATAKANA, FULL-KATAKANA**

This option converts half-width katakana to full-width katakana and is used only with Japanese encoding.

**Restriction** This option cannot be used at the same time with the full-Katakana, half-Katakana option.

**FULL-KATAKANA, HALF-KATAKANA**

This option converts full-width katakana to half-width katakana and is used only with Japanese encoding.

**Restriction** This option cannot be used at the same time with the half-Katakana, full-Katakana option.
KATAKANA, ROMAJI
This option converts the katakana character string to a romaji character string and is used only with Japanese encoding.

Restriction This option cannot be used at the same time with the romaji, katakana option.

ROMAJI, KATAKANA
This option converts the romaji character string to a katakana character string and is used only with Japanese encoding.

Restriction This option cannot be used at the same time with the katakana, romaji option.

FULL-ALPHABET, HALF-ALPHABET
This option converts the Full-Alphabet characters to Half-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.

Restriction This option cannot be used at the same time with the Half-Alphabet, Full-Alphabet option.

HALF-ALPHABET, FULL-ALPHABET
This option converts the Half-Alphabet characters to Full-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.

Restriction This option cannot be used at the same time with the Full-Alphabet, Half-Alphabet option.

LOWERCASE, UPPERCASE
This option converts lowercase alphabet characters to uppercase alphabet characters.

Restriction This option cannot be used at the same time with the Uppercase, Lowercase option.

UPPERCASE, LOWERCASE
This option converts uppercase alphabet characters to lowercase alphabet characters.

Restriction This option cannot be used at the same time with the Lowercase, Uppercase option.

PROPER
This option specifies the following default options based on the encoding:

- Japanese encoding
- Half-Katakana, Full-Katakana
- Full-alphabet, Half-alphabet
- Lowercase, Uppercase
- Korean encoding:
- Full-alphabet, Half-alphabet
- Chinese encoding:
- Full-alphabet, Half-alphabet
- Taiwanese encoding:
Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

This function converts the input string based on the specified options and default options. The KPROPCASE function supports the Chinese, Japanese, Korean, Taiwanese (CJKT) environment.

Example

The following example demonstrates the functionality of the KPROPCASE function:

```sas
length fullkana halfkana upper lower fullalpha $ 200;
length str1 str2 str3 str4 str5 str7 str8 $ 30 str6 $44;
lower = 'do-naxtutsu'; /* Doughnuts in Japanese Roman word. */
upper = 'DO-NAXTUTSU'; /* Doughnuts in Japanese Roman word. */
fullkana = unicode('\u30C9\u30FC\u30CA\u30C3\u30C4');
halfkana = unicode('\uff84\uff9e\uff70\uff85\uff6f\uff82');
fullalpha = unicode('\uff24\uff2f\uff0d\uff2e\uff21\uff38\uff34\uff35\uff33\uff35');

str1 = kpropcase(fullkana, 'full-katakana,half-katakana');
if (halfkana EQ trim(str1)) then
  put str1= $hex14.;

str2 = kpropcase(halfkana, 'half-katakana, full-katakana');
if (fullkana EQ trim(str2)) then
  put str2= $hex22.;

str3 = kpropcase(fullkana, 'katakana,romaji');
if (trim(str3) EQ upper) then
  put str3= ;

str4 = kpropcase(upper, 'romaji,katakana');
if (trim(str4) EQ fullkana) then
  put str4= $hex22.;

str5 = kpropcase(fullalpha, 'full-alphabet, half-alphabet');
if (trim(upper) EQ str5) then
  put str5=;

str6 = kpropcase(upper, 'half-alphabet, full-alphabet');
if (trim(str6) EQ fullalpha) then
  put str6= $hex46.;

str7 = kpropcase(lower, 'lowercase, uppercase');
if (trim(str7) EQ upper) then
  put str7=;

str8 = kpropcase(upper, 'uppercase, lowercase');
if (trim(str8) EQ lower) then
  put str8=;

RESULTS:
str1=C4DEB0C5AFC220
str2=8368815B83698362836320
str3=DO-NAXTUTSU
str4=8368815B83698362836320
str5=DO-NAXTUTSU
str6=8263826E817C826D826082778273827482738272827420
str7=DO-NAXTUTSU
str8=do-naxtutsu
```

□ Full-alphabet, Half-alphabet
KPROPCHAR Function

Converts Unicode to the corresponding characters enclosed in parenthesis.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\( str = \text{KPROPCHAR}(<\text{instr}>) \)

Required Arguments

- **str**: result string. Unicode is converted to characters.
- **instr**: input data string.

Details

This function converts Unicode such as \( \text{①} \) (0x2460) to an equivalent character (1). The KPROPCHAR function converts the characters from the following ranges:

- Enclosed alphanumeric values: \( \text{①} \) to \( \text{⑱} \). See http://www.unicode.org/charts/PDF/U2460.pdf.
- Dingbats: \( \text{⑬} \) to \( \text{⑱} \). See http://www.unicode.org/charts/PDF/U2700.pdf.
- Enclosed CJK letters and months: \( \text{⑴} \) to \( \text{⑳} \). See http://www.unicode.org/charts/PDF/U3200.pdf.

Example

The following example demonstrates the functionality of the KPROPCHAR function. These examples are processed in a UTF8 SAS session:

```sas
length in1 out1 $30 ;
  in1=unicode('\u2460\u2473\u277F\u325F');
  out1=KPROPCHAR(in1);
  put out1;
RESULTS:
(1) (20) (-10) (35)
```
### KPROPDATA Function

Removes or converts unprintable characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

**Syntax**

\[ str = \text{KPROPDATA}(\langle instr \rangle, \langle options \rangle, \langle input encode name \rangle, \langle output encode name \rangle) \]

**Required Arguments**

- **str**
  - specifies a data string that has been converted and is in the session encoding.

- **instr**
  - specifies the input data string.

- **options**
  - specifies instructions on processing unprintable characters:
    - **UESC**
      - Converts unprintable characters using a Unicode escaped string (for example, \u0000\u1234).
TRIM
   Removes unprintable characters. No replacement character is used.

BLANK or ' '
   Replaces each unprintable character with a single-byte blank.

QUESTION or '?'
   Replaces unprintable characters with a single-byte '?'.

HEX
   Replaces unprintable characters with a hexadecimal representation (for example, 0x810x82).

TRUNCATE or TRUNC
   Truncates the data string when the first unprintable character is encountered.

REMOVE
   Removes the data string if any unprintable characters are found.

NCR
   Encodes the unprintable characters using NCR representation if the code is available in Unicode.

PUNC
   Note: This function runs in SAS Viya 3.5 and is not supported in SAS 9.

Converts commonly used 8-bit punctuation marks to 7-bit characters.

WLATIN1 characters are 1-byte, but some of these characters are 2-4 bytes in UTF-8. This option converts punctuation marks that are 1-byte in WLATIN1, but are 2-4 bytes in UTF-8. Here are the punctuation marks that can be converted:
### Table 13.7  Smart Quotation Marks and Punctuation Characters

<table>
<thead>
<tr>
<th>Category</th>
<th>Punctuation character</th>
<th>Description</th>
<th>WLATIN1 code point (Hex)</th>
<th>UTF-8 code point (Hex)</th>
<th>ASCII Character</th>
<th>Description</th>
<th>ASCII code point (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quotation marks</td>
<td>’</td>
<td>single low-9 quotation mark</td>
<td>82</td>
<td>E2 80 9A</td>
<td>,</td>
<td>comma</td>
<td>2C</td>
</tr>
<tr>
<td></td>
<td>“</td>
<td>double low-9 quotation mark</td>
<td>84</td>
<td>E2 80 9E</td>
<td>”</td>
<td>quotation mark</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>’</td>
<td>left single quotation mark</td>
<td>91</td>
<td>E2 80 98</td>
<td>’</td>
<td>apostrophe</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>‘</td>
<td>right single quotation mark</td>
<td>92</td>
<td>E2 80 99</td>
<td>’</td>
<td>apostrophe</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>“</td>
<td>left double quotation mark</td>
<td>93</td>
<td>E2 80 9C</td>
<td>”</td>
<td>quotation mark</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>”</td>
<td>right double quotation mark</td>
<td>94</td>
<td>E2 80 9D</td>
<td>”</td>
<td>quotation mark</td>
<td>22</td>
</tr>
<tr>
<td>Angle brackets</td>
<td>‹</td>
<td>single left-pointing angle quotation mark</td>
<td>8B</td>
<td>E2 80 B9</td>
<td>&lt;</td>
<td>less-than sign</td>
<td>3C</td>
</tr>
<tr>
<td></td>
<td>›</td>
<td>single right-pointing angle quotation mark</td>
<td>9B</td>
<td>E2 80 BA</td>
<td>&gt;</td>
<td>greater-than sign</td>
<td>3E</td>
</tr>
<tr>
<td>Ellipsis</td>
<td>…</td>
<td>horizontal ellipsis</td>
<td>85</td>
<td>E2 80 A6</td>
<td>…</td>
<td>full stop</td>
<td>2E 2E 1</td>
</tr>
<tr>
<td>Bullet</td>
<td>•</td>
<td>bullet</td>
<td>95</td>
<td>E2 80 A2</td>
<td>*</td>
<td>asterisk</td>
<td>2A</td>
</tr>
</tbody>
</table>
input encode name
specifies the input data's encoding name if necessary. If the input encode name is not specified, then the KPROPDATA function processes the data as the current SAS session encoded string. For information about SAS encoding names, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

output encode name
specifies the output data's encoding name. If the encoding name is not specified, the KPROPDATA function recognizes the output as the current SAS session encoding. For information about SAS encoding names, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

Details
This function converts the input data string to the current SAS session encoding and removes or replaces unprintable characters based on the options.

Examples:
Example 1
The following example demonstrates the functionality of the KPROPDATA function:

data test;
  length instr $12;
  length str1 str2 str3 str4 str5 str6 str7 str8 str9 str10$ 50;
  instr = "534153"x||"ae"x || " System";
  put instr;
  str1 = kpropdata(instr);
  put str1= +2 str1= $hex26.;
  str2 = kpropdata(instr,'UESC');
  put str2= +2 str2= $hex26.;
  str3 = kpropdata(instr, 'UESC','wlatin1');
  put str3= +2 str3= $hex26.;
  str4 = kpropdata(instr,'UESC','wlatin1');
  put str4= +2 str4= $hex26.;
  str5 = kpropdata(instr,'?','wlatin1');
  put str5= +2 str5= $hex26.;
  str6 = kpropdata(instr,'?', 'wlatin1');
KREVERSE Function

443

put str6= +2 str6= $hex26.;
str7 = kpropdata(instr,'hex', 'wlatin1');
put str7= +2 str7= $hex26.;
str8 = kpropdata(instr,'TRUNC', 'wlatin1');
put str8= +2 str8= $hex26.;
str9 = kpropdata(instr,'REMOVE', 'wlatin1');
put str9= +2 str9= $hex26.;
str10 = kpropdata(instr,'NCR', 'wlatin1');
put str10= +2 str10= $hex26.;
run;

The preceding statements produce these results:
SAS? System
str1=SAS? System
str1=534153AE2053797374656D2020
str2=SAS? System
str2=534153AE2053797374656D2020
str3=SAS\uff6e System
str3=5341535C75666636652053797374656D20
str4=SAS System
str4=5341532053797374656D202020
str5=SAS System
str5=534153202053797374656D2020
str6=SAS? System
str6=5341533F2053797374656D2020
str7=SAS\xAE System
str7=5341535C784145205379737465
str8=SAS
str8=53415320202020202020202020
str9=
str9=20202020202020202020202020
str10=SAS® System
str10=53415326233137343B20537973

Example 2
This example shows the functionality of the PUNC option. The string assigned to
instr contains several punctuation characters that are not ASCII characters. This
example was run in SAS with a WLATIN1 session encoding.
data _null_;
length instr $12;
length outstr $14;
instr = "—–‘’‚“”„•…‹›"; /* "969791928293948495858B9B"x */
put instr= /instr=hex.;
outstr = kpropdata(instr, 'PUNC');
put outstr= /outstr=hex.;
run;

The preceding statements produce these results:
instr =—–‘’‚“”„•…‹›
instr =979691928293948495858B9B
outstr=--'',"""*...<>
outstr=2D2D27272C2222222A2E2E2E3C3E

KREVERSE Function
Reverses a character expression.
Categories:

CAS
DBCS


Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: Non-DBCS equivalent function is REVERSE in SAS Functions and CALL Routines: Reference.

Syntax

KREVERSE(argument)

Required Argument

argument specifies any SAS character expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>result=kreverse('漢字のテスト');</td>
<td>result=トステの字漢</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KRIGHT Function

Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.

Categories: CAS
            DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: See “RIGHT Function” in SAS Functions and CALL Routines: Reference.

Syntax

KRIGHT(argument)
Required Argument

*argument*

speccifies any SAS character expression.

Details

See "Internationalization Compatibility for SAS String Functions" on page 343 for restrictions and more information.

The KRIGHT function returns an argument with trailing blanks moved to the start of the value. The argument's length does not change.

The KRIGHT, KLEFT and KTRIM functions remove Unicode space characters. See Table 13.30 on page 430 for a list of the Unicode characters.

Example

The following example uses Japanese characters.

```
Statements                              Results

data null;
    a='漢字 の テスト ';                 漢字 の テスト
    b=kright(a);
    put '-------1--------2-----';       漢字 の テスト
    put a $25.;
    put b $25.;
run;
```

See Also

Functions:

- “KCOMPRESS Function” on page 405
- “KLEFT Function” on page 430
- “KTRIM Function” on page 460

KSCAN Function

Selects a specified word from a character expression.

Categories: CAS DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.
Tip: Non-DBCS equivalent function is SCAN in SAS Functions and CALL Routines: Reference.

Syntax

KSCAN(argument,n<, delimiters>)

Required Arguments

argument

specifies any character expression.

n

specifies a numeric expression that produces the number of the word in the character expression that you want KSCAN to select.

Tip If n is negative, KSCAN selects the word in the character expression starting from the end of the string. If |n| is greater than the number of words in the character expression, KSCAN returns a blank value.

delimiters

specifies a character variable that produces characters that you want KSCAN to use as word separators in the character expression.

Defaults If you omit delimiters in an ASCII environment, SAS uses blank . < ( + & / $ *) ; ^ – / , % | . In ASCII environments without the ^ character, KSCAN uses the ~ character instead.

If you omit delimiters on an EBCDIC environment, SAS uses blank . < ( + | & / $ *) ; ¬ – / , % | ¢

Tip If you represent delimiters as a constant, enclose delimiters in quotation marks.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

Leading delimiters before the first word in the character string do not effect KSCAN. If there are two or more contiguous delimiters, KSCAN treats them as one.

The KSCAN function does not have a modify argument. Use the KSCANX function if you need the modify functionality. KSCANX is a complete NLS replacement for the SCAN function.

Example

The following example uses Japanese characters.
KSCANX Function

Selects a specified word from a character expression using a modifier to process the function's action.

Categories: CAS
           DBCS

Restriction: This function is assigned an I18N Level 2 status, and is designed for DBCS data. However, if the first argument, string has single-byte characters, then the KSCANX function processes the SBCS data. For more information, see "Internationalization Compatibility for SAS String Functions" on page 343.

Syntax

\[ \text{KSCANX}(\text{string}, \text{count}, <\text{character-list}>, <\text{modifier}>) \]

Required Arguments

\textit{string}

specifies a character constant, variable, or expression.

See Also

"KSCANX Function" on page 447

```
data _null_;  
  length x $20 y $20;  
  text1='これは漢字関数のテストです。';  
  x='norblank';  
  i=1;  
  do until(x='');  
    x=kscan(text1,i,'はの');  
    put x=;  
    i=i+1;  
  end;  
  y='norblank';  
  i=-1;  
  do until(y='');  
    y=kscan(text1,i,'はの');  
    put y=;  
    i=i-1;  
  end;  
run;  
```
count

is a nonzero numeric constant, variable, or expression that has an integer value. The integer value specifies the number of the word in the character string that you want KSCANX to select. For example, a value of 1 indicates the first word, a value of 2 indicates the second word, and so on. The following rules apply:

- If count is positive, KSCANX counts words from left to right in the character string.
- If count is negative, KSCANX counts words from right to left in the character string.

Optional Arguments

count specifies an optional character expression that initializes a list of characters. This list determines which characters are used as the delimiters that separate words. The following rules apply:

- By default, all characters in count are used as delimiters.
- Specifying a modifier can also change the count-list used as delimiters. For example, if you specify the K modifier in the modifier argument, then all characters that are not in count are used as delimiters.

Note: For more information see “Using Default Delimiters in ASCII and EBCDIC Environments” in SAS Functions and CALL Routines: Reference.

TIP You can add more characters to count by using other modifiers.

modifier

specifies a character constant, variable, or expression in which each non-blank character modifies the action of the KSCANX function. Blanks are ignored. Use the following characters as modifiers:

- a or A adds alphabetic characters to the list of characters.
- b or B scans string from right to left instead of from left to right.
- c or C adds control characters to the list of characters.
- d or D adds digits to the list of characters.
- f or F adds an underscore and English letters (that is, valid first characters in a SAS variable name using VALIDVARNAME=V7) to the list of characters.
- g or G adds graphic characters to the list of characters. Graphic characters are those that, when printed, produce an image on paper.
- h or H adds a horizontal tab to the list of characters.
- i or I ignores the case of the characters.
- k or K causes all characters that are not in the list of characters to be treated as delimiters. That is, if K is specified, then characters that are in the list of characters are kept in the returned value rather than being omitted because they are delimiters. If K is not
specified, then all characters that are in the list of characters are treated as delimiters.

\( l \text{ or } L \) adds lowercase letters to the list of characters.

\( m \text{ or } M \) specifies that multiple consecutive delimiters, and delimiters at the beginning or end of the string argument, refer to words that have a length of zero. If the M modifier is not specified, then multiple consecutive delimiters are treated as one delimiter, and delimiters at the beginning or end of the string argument are ignored.

\( n \text{ or } N \) adds digits, an underscore, and English letters (that is, the characters that can appear in a SAS variable name using VALIDVARNAMES=V7) to the list of characters.

\( o \text{ or } O \) processes the character-list and modifier arguments only once, rather than every time the KSCANX function is called. Using the O modifier in the DATA step (excluding WHERE clauses), or in the SQL procedure can make KSCANX run faster when you call it in a loop where the character-list and modifier arguments do not change. The O modifier applies separately to each instance of the KSCANX function in your SAS code, and does not cause all instances of the KSCANX function to use the same delimiters and modifiers.

\( p \text{ or } P \) adds punctuation marks to the list of characters.

\( q \text{ or } Q \) ignores delimiters that are inside substrings that are enclosed in quotation marks. If the value of the string argument contains unmatched quotation marks, then scanning from left to right produces different words than scanning from right to left.

\( r \text{ or } R \) removes leading and trailing blanks from the word that KSCANX returns. If you specify the Q and R modifiers, the KSCANX function first removes leading and trailing blanks from the word. Then, if the word begins with a quotation mark, KSCANX also removes one layer of quotation marks from the word.

\( s \text{ or } S \) adds space characters to the list of characters (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed).

\( t \text{ or } T \) trims trailing blanks from the string and character-list arguments. If you want to remove trailing blanks from just one character argument instead of both character arguments, then use the TRIM function instead of the KSCANX routine with the T modifier.

\( u \text{ or } U \) adds uppercase letters to the list of characters.

\( w \text{ or } W \) adds printable (writable) characters to the list of characters.

\( x \text{ or } X \) adds hexadecimal characters to the list of characters.

**TIP** If the modifier argument is a character constant, enclose the argument in quotation marks. Specify multiple modifiers in a single set of quotation marks. A modifier argument can also be expressed as a character variable or expression.
Details

Delimiter and Word

A delimiter is any of several characters that are used to separate words. You can specify the delimiters in the character-list and modifier arguments.

If you specify the Q modifier, delimiters inside substrings that are enclosed in quotation marks are ignored.

In the KSCANX function, word refers to a substring that has all of these characteristics:

- is bounded on the left by a delimiter or the beginning of the string
- is bounded on the right by a delimiter or the end of the string
- A word can have a length of zero if there are delimiters at the beginning or end of the string, or if the string contains two or more consecutive delimiters. However, the KSCANX function ignores words that have a length of zero unless you specify the M modifier.

Using Default Delimiters in ASCII and EBCDIC Environments

If you use the KSCANX function with only two arguments, then the default delimiters depend on whether your computer uses ASCII or EBCDIC characters.

- If your computer uses ASCII characters, the default delimiters are:
  \[ \text{blank, !, $, %, &}, (, )*, +, -, ., /, ;, <, ^, |} \]
  In ASCII environments that do not contain the ^ character, the KSCANX function uses the ~ character instead.

- If your computer uses EBCDIC characters, then the default delimiters are as follows:
  \[ \text{blank, !, $, %, &}, (, )*, +, -, ., /, ;, <, ¬, |} \]
  If you use the modifier argument without specifying any characters as delimiters, then the only delimiters that are used are delimiters that are defined by the modifier argument. In this case, the lists of default delimiters for ASCII and EBCDIC environments are not used. In other words, modifiers add to the list of delimiters that are explicitly specified by the character-list argument. Modifiers do not add to the list of default modifiers.

Using the KSCANX Function with the M Modifier

If you specify the M modifier, the number of words in a string is defined as one plus the number of delimiters in the string. However, if you specify the Q modifier, delimiters that are inside quotation marks are ignored.

If you specify the M modifier, the KSCANX function returns a word with a length of zero if one of these conditions is true:

- The string begins with a delimiter and you request the first word.
- The string ends with a delimiter and you request the last word.
- The string contains two consecutive delimiters and you request the word that is between the two delimiters.
Using the KSCANX Function without the M Modifier

If you do not specify the M modifier, the number of words in a string is defined as the number of maximal substrings of consecutive non-delimiters. However, if you specify the Q modifier, delimiters that are inside quotation marks are ignored.

If you do not specify the M modifier, the KSCANX function acts in these ways:

- ignores delimiters at the beginning or end of the string
- treats two or more consecutive delimiters as if they were a single delimiter

If the string contains no characters other than delimiters, or if you specify a count that is greater in absolute value than the number of words in the string, then the KSCANX function returns one of the following items:
  - a single blank when you call the KSCANX function from a DATA step
  - a string with a length of zero when you call the KSCANX function from the macro processor.

Using Null Arguments

This example scans a string for the first and last words:

- A negative count instructs the KSCANX function to scan from right to left.
- Leading and trailing delimiters are ignored because the M modifier is not used.
- In the last observation, all characters in the string are delimiters.

Example

This example uses Chinese characters:

```sas
data _null_
  xyz='SAS是全球最大的软件公司之一。sas是全球商业智能和分析软件与服务领袖。'
  substr=kscan(xyz, 2, ' ', 'sp') /* Use spaces and punctuations as delimiter. */
  put substr=; run;
```

KSTRCAT Function

Concatenates two or more character expressions.

Categories: CAS
            DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Tip: Non-DBCS equivalent function is CAT in SAS Functions and CALL Routines: Reference.
Syntax

KSTRCAT(argument-1, argument-2<, … argument-n>)

Required Argument

argument
specifies any single-byte or double-byte character expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

If the KSTRCAT function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

In the SQL procedure, or in a WHERE clause in any procedure, the maximum length of a word that is returned by the KSTRCAT function is 200 characters.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data _null;</td>
<td></td>
</tr>
<tr>
<td>text1='漢字';</td>
<td></td>
</tr>
<tr>
<td>text2='文字列';</td>
<td></td>
</tr>
<tr>
<td>text3='連結';</td>
<td></td>
</tr>
<tr>
<td>result=kstrcat(text1,text2,'の',text3);</td>
<td>result=漢字文字列の連結</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KSTRIP Function

Removes leading and trailing blanks from a character string.

Categories: CAS
Character

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Note: This function supports the VARCHAR data type.
Syntax

\texttt{KSTRIP(string)}

Required Argument

\textit{string}

specifies a character constant, variable, or expression.

Details

The \texttt{KSTRIP} function returns the argument with all leading and trailing SBCS blanks removed. If the argument is blank, \texttt{KSTRIP} returns a string with a length of zero.

Assigning the results of \texttt{KSTRIP} to a variable does not affect the length of the receiving variable. If the value that is trimmed is shorter than the length of the receiving variable, SAS pads the value with new trailing blanks.

In a DATA step, if the \texttt{KSTRIP} function returns a value to a variable that has not previously been assigned a length, then that variable is given the length of the argument.

Comparisons

The \texttt{KSTRIP} and \texttt{STRIP} functions return a string with a length of zero, for strings that are blank, and the \texttt{TRIM} function returns a single blank.

The \texttt{KSTRIP} and \texttt{STRIP} functions truncates the output if the receiving variable length is less than the input string. The \texttt{STRIP} function processes only SBCS string truncation. The \texttt{KSTRIP} function processes SBCS and DBCS data.

The \texttt{KSTRIP} and \texttt{STRIP} functions normalize data by removing unnecessary blanks. The \texttt{KSTRIP} function should be used in a DBCS environment. The \texttt{KSTRIP} and \texttt{STRIP} functions behave similarly. The \texttt{KTRIM} and \texttt{KLEFT} functions can also be used for data normalization. However, the \texttt{KTRIM} and \texttt{KLEFT} functions remove all types of DBCS spaces and can cause performance problems, as in this code example:

\begin{verbatim}
long_char_variable = KTRIM(KLEFT(long_char_variable));
\end{verbatim}

The \texttt{KSTRIP} function resolves these issues. You can modify the code using the \texttt{KSTRIP} function to successfully perform DBCS data normalization in a DBCS SAS session using this code:

\begin{verbatim}
long_char_variable = KSTRIP(long_char_variable);
\end{verbatim}

Example

The following example shows how the \texttt{KSTRIP} function deletes leading and trailing blanks, and how the DBCS character is truncated. This example uses the Japanese Shift_JIS encoding.
### Statements

```sas
data _null_;  
  length a $5;  
  v = unicode('"\u0020\u0020\u8404\u8404\u8404\u0020\u0020"');  
  put v = hex.;  
  a = kstrip(v);  
  put a = hex.;  
run;
```

### Results

```
93B893B820
```

### See Also

**Function**
- "STRIP Function" in *SAS Functions and CALL Routines: Reference*

### KSUBSTR Function

Extracts a substring from an argument.

**Categories:**
- CAS
- DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see *Internationalization Compatibility* on page 343.

**Tip:**
See “SUBSTR (right of =) Function” in *SAS Functions and CALL Routines: Reference*.

### Syntax

```
KSUBSTR(argument, position<n> )
```

**Required Arguments**

- **argument**
  - specifies any SAS character expression.

- **position**
  - specifies a numeric expression that is the beginning character position.

- **n**
  - specifies a numeric expression that is the length of the substring to extract.
Interaction

If $n$ is larger than the length of the expression that remains in argument after position, SAS extracts the remainder of the expression.

Tip

If you omit $n$, SAS extracts the remainder of the expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The KSUBSTR function returns a portion of an expression that you specify in argument. The portion begins with the character specified by position and is the number of characters specified by $n$.

A variable that is created by KSUBSTR obtains its length from the length of argument.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>text='漢字文字列の抽出';</td>
<td></td>
</tr>
<tr>
<td>result=kssubstr(text,1,2);</td>
<td>result=漢字</td>
</tr>
<tr>
<td>put result=;</td>
<td>result=文字列の</td>
</tr>
<tr>
<td>result=kssubstr(text,3,4);</td>
<td>result=抽出</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>kstart=7;</td>
<td></td>
</tr>
<tr>
<td>klen=2;</td>
<td></td>
</tr>
<tr>
<td>result=kssubstr(text,kstart,klen);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Function:

- “KSUBSTR Function” on page 455

KSUBSTRB Function

Extracts a substring from an argument according to the byte position of the substring in the argument.

Category: DBCS
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\texttt{KSUBSTRB(argument, position<n> )}

Required Arguments

- \textit{argument}: specifies any SAS character expression.
- \textit{position}: specifies the beginning character position in byte units.
- \textit{n}: specifies the length of the substring to extract in byte units.

Interaction

If \textit{n} is larger than the length (in byte units) of the expression that remains in \textit{argument} after \textit{position}, SAS extracts the remainder of the expression.

Tip

If you omit \textit{n}, SAS extracts the remainder of the expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The \texttt{KSUBSTRB} function returns a portion of an expression that you specify in \textit{argument}. The portion begins with the byte unit specified by \textit{position} and is the number of byte units specified by \textit{n}.

A variable that is created by \texttt{KSUBSTRB} obtains its length from the length of \textit{argument}.

Examples:

Example 1

The following examples use Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>text=`漢字文字列の抽出';</td>
<td>result=漢字文</td>
</tr>
<tr>
<td>result=ksubstrb(text,1,6);</td>
<td>result=字文</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>result=ksubstrb(text,3,5);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
Example 2

If the position or $n$ arguments do not align at the character boundary, broken characters are converted to blanks. In this example, the broken characters become 0x20 (SBCS blank).

```plaintext
data _null_
length sub ksub ksubb $6;
str = '中国语';
sub = substr(str, 2, 6);
ksub = ksubstr(str, 2, 6);
ksubb = ksubstrb(str, 2, 6);
put str= / str= $hex16.;
put sub= / sub= $hex12.;
put ksub= / ksub= $hex12.;
put ksubb= / ksubb= $hex12.;
run;
```

These statements produce these results:

```plaintext
str = 中国语
str = E4B8ADE5BBDE8AFAD
sub = 国
sub = B8ADE5BBDE8
ksub = 国语
ksub = E5BBDE8AFAD
ksubb = 国
ksubb = 2020E5BBD20
```

See Also

**Function:**

- “KSUBSTR Function” on page 454

---

**KSUBSTRN Function**

Returns a substring, allowing a result with a length of zero.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8).

**See:** “SUBSTRN Function” in *SAS Functions and CALL Routines: Reference* and “Internationalization Compatibility for SAS String Functions” on page 343 for more information.
Syntax

KSUBSTRN(string, position, <length>)

Required Arguments

string
specifies a character or numeric constant, variable, or expression.

If string is numeric, then it is converted to a character value that uses the BEST32 format. Leading and trailing blanks are removed, and no message is sent to the SAS log.

position
is an integer that specifies the position of the first character in the substring.

Optional Argument

length
is an integer that specifies the length of the substring. If you do not specify length, the KSUBSTRN function returns the substring that extends from the position that you specify to the end of the string.

Details

The functionality of KSUBSTRN is similar to SUBSTRN. The difference is that if the first parameter is numeric, SUBSTRN’s output length is 8, and KSUBSTRN’s output length is 200.

Example

This example shows the different functionality between the functions KSUBSTR and KSUBSTRN.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>ksubstr_result = &quot;*&quot;</td>
<td></td>
</tr>
<tr>
<td>put ksubstr_result=;</td>
<td></td>
</tr>
<tr>
<td>ksubstr_result = &quot;*&quot;</td>
<td></td>
</tr>
<tr>
<td>put ksubstr_result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>ksubstr_result=* 1234*</td>
<td></td>
</tr>
<tr>
<td>ksubstr_result=<em>234.56</em></td>
<td></td>
</tr>
</tbody>
</table>

KTRANSLATE Function

Replaces specific characters in a character expression.

Categories: CAS
            DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.
KTRANSLATE Function

**z/OS specifics:** This function runs only on z/OS.
Uses the EBCDIC code sequence.

**Tip:** Non-DBCS equivalent function is **TRANSLATE** in *SAS Functions and CALL Routines: Reference*.

---

**Syntax**

**KTRANSLATE**(*source,* to-1,* from-1<,...,*to-n,* from-n>*)

**Required Arguments**

- **source** specifies the SAS expression that contains the original character value.
- **to** specifies the characters that you want KTRANSLATE to use as substitutes.
- **from** specifies the characters that you want KTRANSLATE to replace.

**Interaction** Values of **to** and **from** correspond on a character-by-character basis; KTRANSLATE changes character one of **from** to character one of **to**, and so on. If **to** has fewer characters than **from**, KTRANSLATE changes the extra **from** characters to blanks. If **to** has more characters than **from**, KTRANSLATE ignores the extra **to** characters.

**Note** You must have pairs of **to** and **from** arguments on some operating environments. On other operating environments, a segment of the collating sequence replaces null **from** arguments.

---

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

You can use KTRANSLATE to translate a single-byte character expression to a double-byte character expression or to translate a double-byte character expression to a single-byte character expression.

The maximum number of **to** and **from** character pairs that KTRANSLATE accepts should not exceed 256. There is no functional difference between using many pairs of shorter arguments and using fewer pairs of longer arguments. In the following example, each line has a different argument and character combination pair than the other, but they produce the same output.

The first line in Example 1 has two argument pairs and three character pairs:

```
x=ktranslate('AABBAABABB','12','AB','3','C');
```

The second line in Example 1 has one argument pair and three character pairs:

```
y=ktranslate('AABBAABABB','123','ABC');
```

If the KTRANSLATE function returns a value to a variable that has not yet been assigned a length, the variable is assigned a default length of 200.
Examples:

Example 1
This example shows the to and from character pairs.

```sas
data test;
  x=ktranslate('AABBAABABB','12','AB','3','C'); /* Two to/from argument pairs, three from/to char pairs. */
  y=ktranslate('AABBAABABB','123','ABC'); /* One to/from argument pair, three from/to char pairs. */
  put x=;
  put y=;
run;
```

SAS writes the following output to the log:

```
x=1122112122
y=1122112122
```

Example 2
The following example uses Japanese characters.

```sas
data _null_
  result = ktranslate('伝等', '中忠', '伝伝');
  put result=;
run;
```

result=中等

---

**KTRIM Function**

Removes trailing DBCS blanks and SO/SI from character expressions.

**Categories:**
- CAS
- DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see [Internationalization Compatibility on page 343](#).

**Tip:**
Non-DBCS equivalent function is “TRIM Function” in [SAS Functions and CALL Routines: Reference](#).

**Syntax**

```
KTRIM(argument)
```
Required Argument

*argument*

specifies any SAS character expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

KTRIM copies a character argument, removes all trailing blanks, and returns the trimmed argument as a result. If the argument is blank, KTRIM returns one blank. KTRIM is useful for concatenating because concatenation does not remove trailing blanks.

Note: If the argument is blank, KTRIM returns one blank for the CHAR type. If the argument is blank, KTRIM returns an empty blank or 0 value for the varchar type. The varchar type uses character semantics, and the CHAR type uses byte semantics.

Assigning the results of KTRIM to a variable does not affect the length of the receiving variable. If the trimmed value is shorter than the length of the receiving variable, SAS pads the value with new blanks as it assigns it to the variable.

The KRIGHT, KLEFT and KTRIM functions remove Unicode space characters. See Table 13.30 on page 430 for a list of the Unicode characters.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>漢字の テスト</td>
</tr>
<tr>
<td>part1='漢字の';</td>
<td>漢字のテスト</td>
</tr>
<tr>
<td>part2='テスト';</td>
<td></td>
</tr>
<tr>
<td>hasblank=part1</td>
<td></td>
</tr>
<tr>
<td>noblank=ktrim(part1)</td>
<td></td>
</tr>
<tr>
<td>put hasblank;</td>
<td></td>
</tr>
<tr>
<td>put noblank;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:

- “KCOMPRESS Function” on page 405
- “KLEFT Function” on page 430
- “KRIGHT Function” on page 444
KTRUNCATE Function

Truncates a string to a specified length in byte unit without breaking multibyte characters.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\[ \text{KTRUNCATE}(\text{argument, number, length}) \]

Required Arguments

- **argument**: specifies any SAS character expression.
- **number**: is numeric.
- **length**: is an integer.

Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The KTRUNCATE function truncates a full-length `number` (stored as a double) to a smaller number of bytes, as specified in `length` and pads the truncated bytes with 0s. The truncation and subsequent expansion duplicate the effect of storing numbers in less than full length and then reading them.

Example

The following example uses Japanese characters.

```
data _null_;  
x1=ktruncate(’漢字のテスト’,4);  
x2=ktruncate(’漢字のテスト’,5);  
x3=ktruncate(’漢字のテスト’,6);  
y1=ktruncate(’漢字のテスト’,3,6);  
y2=ktruncate(’漢字のテスト’,4,6);  
y3=ktruncate(’漢字のテスト’,5,6);  
put x1= / x2= / x3= / y1= / y2= / y3=;  
run;```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| data _null_;  
x1=ktruncate(’漢字のテスト’,4);  
x2=ktruncate(’漢字のテスト’,5);  
x3=ktruncate(’漢字のテスト’,6);  
y1=ktruncate(’漢字のテスト’,3,6);  
y2=ktruncate(’漢字のテスト’,4,6);  
y3=ktruncate(’漢字のテスト’,5,6);  
put x1= / x2= / x3= / y1= / y2= / y3=;  
run; | x1=漢字  
x2=漢字  
x3=漢字の  
y1=字のテ  
y2=のテ  
y3=のテス |
KUPCASE Function

Converts the lowercase alphabetic letters to uppercase letters.

Categories: CAS
DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.


Syntax

KUPCASE(argument)

Required Argument

argument

specifies any SAS character expression.

Details

The KUPCASE function copies a character argument, converts the lowercase alphabetic letters to uppercase letters, and returns the altered value.

The definitions of the alphabetic letters differ based on the SAS session encoding. Here are typical SAS sessions and how KUPCASE processes alphabetic letters for the different sessions.

UTF-8 (SAS Unicode server)

When the SAS session encoding is UTF-8, the Unicode character attributes apply. If a character is classified as a letter and has an uppercase equivalent, KUPCASE returns the uppercase letter. Otherwise, the same letter is returned.

DBCS (For example, Chinese, Japanese, and Korean languages)

KUPCASE converts lowercase letters in the English alphabet to uppercase in a DBCS SAS session.

SBCS (For example, English or European languages)

When the SAS session encoding is SBCS, the TRANTAB option is updated to use the mapping tables for uppercase letters and the lowercase equivalents. The mapping tables for uppercase letters use the naming pattern of nnnn_ucs, where nnnn matches the short encoding name that is documented in SAS National Language Support (NLS): Reference Guide. The KUPCASE function uses these tables to convert the lowercase letters to the uppercase letters that are supported by the session encoding.
Examples:

Example 1

In this example, the text contains a mix of English alphabetic lowercase letters plus the lowercase ñ. The SAS session encoding is Windows code page 1252, which is WLATIN1. Notice that KUPCASE returns all the letters in uppercase, including the Ñ. This example demonstrates results from KUPCASE when the SAS encoding is SBCS and supports Western European characters. The same result is returned for UTF-8 text when the SAS encoding is UTF-8.

```sas
options locale=en_US;
data one;
x='niño';
y=kupcase(x);

put x=;
put y=;
run;
```

```
x=niño
y=NIÑO
```

Example 2

This example uses a mix of Japanese and English alphabetic lowercase letters. In the result from KUPCASE, the English alphabetic letters are converted to uppercase and the Japanese characters are unchanged. The example demonstrates results from KUPCASE when the SAS session uses a DBCS encoding that supports Japanese characters or UTF-8.

```sas
data _null_;  
result=kupcase('abcあいうえお');  
put result=;  
run;  

result=ABCあいうえお
```

---

**KUPDATE Function**

Inserts, deletes, and replaces character value contents.

**Categories:**
CAS
DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.
Syntax

\texttt{KUPDATE(\textit{argument}, \textit{position}, \textit{n}<, \textit{characters-to-replace}> )}
\texttt{KUPDATE(\textit{argument}, \textit{position}<\textit{n}>, \textit{characters-to-replace})}

Required Arguments

\textit{argument} specifies a character variable.

\textit{position} specifies a numeric expression that is the beginning character position.

\textit{n} specifies a numeric expression that is the length of the substring to be replaced.

Restrictions \textit{n} cannot be larger than the length of the expression that remains in \textit{argument} after \textit{position}.

\textit{n} is optional, but you cannot omit both \textit{n} and \textit{characters-to-replace} from the function.

Tip If you omit \textit{n}, SAS uses all of the characters in \textit{characters-to-replace} to replace the values of \textit{argument}.

\textit{characters-to-replace} specifies a character expression that replaces the contents of \textit{argument}.

Restriction \textit{characters-to-replace} is optional, but you cannot omit both \textit{characters-to-replace} and \textit{n} from the function.

Tip Enclose a literal string of characters in quotation marks.

Details

See "Internationalization Compatibility for SAS String Functions" on page 343 for restrictions and more information.

The \texttt{KUPDATE} function replaces the value of \textit{argument} with the expression in \textit{characters-to-replace} starting at the character that you specify in \textit{position}.

If the \texttt{KUPDATE} function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

In the SQL procedure, or in a \texttt{WHERE} clause in any procedure, the maximum length of a word that is returned by the \texttt{KUPDATE} function is 200 characters.

Examples:

Example 1

The following example shows the difference between \texttt{KUPDATE} and \texttt{KUPDATES}.
Statements | Results
---|---
data _null_; 
str='123456';  
x1=str; substr(x1,2,3)="";  
y1=kupdate(str,2,3);  
z1=kupdates(str,2,3);  
put x1= / y1= / z1=;  
x2=str; substr(x2,2,3)="abcd";  
y2=kupdate(str,2,3,'abcd');  
z2=kupdates(str,2,3,'abcd');  
put x2= / y2= / z2=;  
x3=str; substr(x3,2,3)="ab";  
y3=kupdate(str,2,3,'ab');  
z3=kupdates(str,2,3,'ab');  
put x3= / y3= / z3=;  
run;

\[x1=156\]
\[y1=156\]
\[z1=156\]
\[x2=1abc56\]
\[y2=1abc56\]
\[z2=1abc56\]
\[x3=1ab56\]
\[y3=1ab56\]
\[z3=1ab56\]

Example 2

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
data _null_; 
x1=kupdate('感じのテスト',1,2,'漢字');  
x2=kupdate(x1,1,2,'かな');  
x3=kupdate(x1,1,3);  
x4=kupdate(x1,3,'かんじ');  
put x1= / x2= / x3= / x4=;  
run;

\[x1=漢字のテスト\]
\[x2=かなのテスト\]
\[x3=テスト\]
\[x4=漢字かんじ\]

See Also

Functions:

- “KUPDATES Function” on page 468
- “KUPDATEB Function” on page 466

KUPDATEB Function

Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.
Syntax

**KUPDATEB** (*argument, position, n<,characters-to-replace> *)

Required Arguments

*argument*
- specifies a character variable.

*position*
- specifies the beginning character position in byte units.

*n*
- specifies the length of the substring to be replaced in byte units.

Restrictions

*n* cannot be larger than the length (in bytes) of the expression that remains in *argument* after *position*.

*n* is optional, but you cannot omit both *n* and *characters-to-replace* from the function.

Tip
- If you omit *n*, SAS uses all of the characters in *characters-to-replace* to replace the values of *argument*.

*characters-to-replace*
- specifies a character expression to replace the contents of *argument*.

Restriction
- *characters-to-replace* is optional, but you cannot omit both *characters-to-replace* and *n* from the function.

Tip
- Enclose a literal string of characters in quotation marks.

Details

See "Internationalization Compatibility for SAS String Functions" on page 343 for restrictions and more information.

The KUPDATEB function replaces the value of *argument* with the expression in *characters-to-replace*. KUPDATEB replaces *n* byte units starting at the byte unit that you specify in *position*.

If the KUPDATEB function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

In the SQL procedure, or in a WHERE clause in any procedure, the maximum length of a word that is returned by the KUPDATEB function is 200 characters.

Example

The following example uses Japanese characters.
KUPDATES Function

Inserts, deletes, and replaces character value contents.

**Categories:**
CAS
DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see [Internationalization Compatibility on page 343](#).

**Syntax**

\[
\text{KUPDATES}(\text{argument}, \text{position}, \text{n} <, \text{characters-to-replace}> )
\]

\[
\text{KUPDATES}(\text{argument}, \text{position} <, \text{n}>, \text{characters-to-replace})
\]

**Required Arguments**

- **argument**
  specifies a character variable.

- **position**
  specifies a numeric expression that is the beginning character position.

**Optional Arguments**

- **n**
  specifies a numeric expression that is the length of the substring to be replaced.

**Restrictions**

- \( n \) cannot be larger than the length of the expression that remains in \( \text{argument} \) after \( \text{position} \).
n is optional, but you cannot omit both n and characters-to-replace from the function.

Tip
If you omit n, SAS uses all of the characters in characters-to-replace to replace the values of argument.

characters-to-replace
specifies a character expression that replaces the contents of argument.

Restriction
characters-to-replace is optional, but you cannot omit both characters-to-replace and n from the function.

Tip
Enclose a literal string of characters in quotation marks.

Details
See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The KUPDATES function replaces the value of argument with the expression in characters-to-replace. KUPDATES replaces n characters starting at the character that you specify in position.

If the KUPDATES function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

In the SQL procedure, or in a WHERE clause in any procedure, the maximum length of a word that is returned by the KUPDATES function is 200 characters.

Examples:
Example 1
The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>x1=kupdates(’感じるテスト’,1,2,’汉字’);</td>
<td>x1=汉字のテスト</td>
</tr>
<tr>
<td>x2=kupdates(x1,1,2,’kanji’);</td>
<td>x2=」のテスト</td>
</tr>
<tr>
<td>x3=kupdates(x1,1,3);</td>
<td>x3=テスト</td>
</tr>
<tr>
<td>x4=kupdates(x1,3,’かんじ’);</td>
<td>x4=漢字かんじ</td>
</tr>
<tr>
<td>put x1= / x2= / x3= / x4=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

Example 2
The following example shows the difference between KUPDATE and KUPDATES.
### Statements

```sas
data _null_;  
str='123456';  
x1=str; substr(x1,2,3)='';  
y1=kupdate(str,2,3);  
z1=kupdates(str,2,3);  
put x1= / y1= / z1=;  

x2=str; substr(x2,2,3)='abcd';  
y2=kupdate(str,2,3,'abcd');  
z2=kupdates(str,2,3,'abcd');  
put x2= / y2= / z2=;  

x3=str; substr(x3,2,3)='ab';  
y3=kupdate(str,2,3,'ab');  
z3=kupdates(str,2,3,'ab');  
put x3= / y3= / z3=;  
run;
```

<table>
<thead>
<tr>
<th>x1</th>
<th>y1</th>
<th>z1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1abc56</td>
<td></td>
<td>1abc56</td>
</tr>
<tr>
<td>1abc56</td>
<td></td>
<td>1abc56</td>
</tr>
<tr>
<td>1ab</td>
<td></td>
<td>1ab</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>1abc6</td>
<td></td>
<td>1abc6</td>
</tr>
<tr>
<td>1abc6</td>
<td></td>
<td>1abc6</td>
</tr>
</tbody>
</table>

### See Also

**Functions:**
- "KUPDATE Function" on page 464

### KVERIFY Function

**Returns the position of the first character (character-based value) that is unique to an expression.**

**Categories:**
- CAS
- DBCS

**Restriction:**
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see [Internationalization Compatibility](#) on page 343.

**Tip:**
See “VERIFY Function” in [SAS Functions and CALL Routines: Reference](#)

#### Syntax

**KVERIFY**(*source*, *excerpt-1*, ...*excerpt-n*)

#### Required Arguments

- **source**
  - specifies any SAS character expression.

- **excerpt-1, ...excerpt-n**
  - specifies any SAS character expression.

#### Tips
- Enclose a literal string of characters in quotation marks.
- If you specify more than one excerpt, separate them with a comma.
Details

See “Internationalization Compatibility for SAS String Functions” on page 343 for restrictions and more information.

The KVERIFY function returns the position of the first character in source that is not present in any excerpt. If KVERIFY finds every character in source in at least one excerpt, it returns a value of 0.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; input grade :$2. @@; check='良否'; x=kverify(grade,check); if x gt 0 then put 'Invalid grade value: ' grade; cards; 良 否 良 否 良 否 可</td>
<td></td>
</tr>
</tbody>
</table>

KVERIFYB Function

Returns the position of the first character (byte-based value) that is unique to an expression.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

KVERIFYB(source, excerpt-1,<excerpt-n>)

Required Arguments

- **source**
  - specifies any SAS character expression.

- **excerpt**
  - specifies any SAS character expression.

**TIP** If you specify more than one excerpt, separate them with a comma.
Details

KVERIFYB returns the position of the first character in source that is not present in any excerpt. If KVERIFYB finds every character in source in at least one excerpt, it returns a 0.

Comparisons

KVERIFYB returns a byte-based value. KVERIFY returns a character-based value. When processing an SBCS string (for example, in the wlatin1 encoding), these functions return identical results. But, in a DBCS session (for example, EUC-JP or SHIFT-JIS encoding), most CJK characters are 2-byte width, which makes these functions return different results.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>x1=3</td>
</tr>
<tr>
<td>check=‘漢字’;</td>
<td>x2=5</td>
</tr>
<tr>
<td>text=‘漢字の検索’;</td>
<td></td>
</tr>
<tr>
<td>x1=kverify(text,check);</td>
<td></td>
</tr>
<tr>
<td>put x1=;</td>
<td></td>
</tr>
<tr>
<td>x2=kverifyb(text,check);</td>
<td></td>
</tr>
<tr>
<td>put x2=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATE Function

Converts the SAS date value to the date value of the specified locale by using the date format descriptors.

Category: Date and Time

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\[
\text{NLDATE}(date,\text{descriptor})
\]

Required Arguments

\[
\begin{align*}
\text{date} & \quad \text{specifies a SAS date value.} \\
\text{descriptor} & \quad \text{is a variable or expression that specifies how dates and times are formatted in output. The following descriptors are case sensitive:}
\end{align*}
\]
# removes the leading zero from the result.

%% specifies the % character.

%a specifies the short-weekday descriptor. The range for the short-weekday descriptor is Mon–Sun.

%A specifies the long-weekday descriptor. The range for the long-weekday descriptor is Monday–Sunday.

%b specifies the short-month descriptor. The range for the short-month descriptor is Jan–Dec.

%B specifies the long-month descriptor. The range for the long-month descriptor is January–December.

%C specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.

%d specifies the day descriptor and uses 0 padding. The range for the day modifier is 01–31.

%e specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.

%F specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.

%j specifies the day-of-year descriptor as a decimal number and uses a leading zero. The range for the day-of-year descriptor is 1–366.

%m specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.

%o specifies the month descriptor. The range for the month descriptor is 1–12 with blank padding.

%u specifies the weekday descriptor as a number in the range 1–7 that represents Monday–Sunday.

%U specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value using the number of week within the year (Sunday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53 and uses a leading zero and a maximum value of 53.

%V specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value. The number-of-week value is represented as a
decimal number in the range 01–53 and uses a leading zero and a maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year.

`%w` specifies the weekday descriptor as a number in the range 0–6 that represents Sunday–Saturday.

`%W` specifies the week-number-of-year descriptor by calculating the descriptor value as SAS date value by using the number of week within the year (Monday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53 and uses a leading zero and a maximum value of 53.

`%y` specifies the year (2-digit) modifier. The range for the year descriptor is 00–99.

`%Y` specifies the year (4-digit) descriptor. The range for the year descriptor is 1970–2069.

Details
The NLDATE function converts the SAS date value to the date value of the specified locale by using the date descriptors.

Example
The following example shows a log filename that is created from a SAS date value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_Unitedstates;</td>
<td>February-24.log</td>
</tr>
<tr>
<td>logfile=nldate('24Feb2003'd,'%B-%d.log');</td>
<td></td>
</tr>
<tr>
<td>put logfile;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>Februar-24.log</td>
</tr>
<tr>
<td>logfile=nldate('24Feb2003'd,'%B-%d.log');</td>
<td></td>
</tr>
<tr>
<td>put logfile;</td>
<td></td>
</tr>
</tbody>
</table>

The following example shows a weekday name that is created from a SAS date value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_Unitedstates;</td>
<td>474</td>
</tr>
<tr>
<td>logfile=nldate('24Feb2003'd,'%B-%d.log');</td>
<td></td>
</tr>
<tr>
<td>put logfile;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>logfile=nldate('24Feb2003'd,'%B-%d.log');</td>
<td></td>
</tr>
<tr>
<td>put logfile;</td>
<td></td>
</tr>
</tbody>
</table>

----+----1----+

474  Chapter 13 / Dictionary of Functions for NLS
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_unitedstates;</td>
<td>Monday</td>
</tr>
<tr>
<td>weekname=nldate('24Feb2003'd,'%A');</td>
<td></td>
</tr>
<tr>
<td>put weekname;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>Montag</td>
</tr>
<tr>
<td>weekname=nldate('24Feb2003'd,'%A');</td>
<td></td>
</tr>
<tr>
<td>put weekname;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- "NLDATEw. Format" on page 154

### NLDATM Function

Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.

**Category:** Date and Time

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

### Syntax

\[
\text{NLDATM}(\text{datetime},\text{descriptor})
\]

**Required Arguments**

- **datetime**
  - specifies a SAS datetime value.

- **descriptor**
  - is a variable or expression that specifies how dates and times are formatted in output. The following descriptors are case sensitive:

  - `#`
    - removes the leading zero from the result.

  - `%`
    - specifies the % character.

  - `%a`
    - specifies the short-weekday descriptor. The range for the day descriptor is Mon–Sun.
%A specifies the long-weekday descriptor. The range for the long-weekday descriptor is Monday–Sunday.

%b specifies the short-month descriptor. The range for the short-month descriptor is Jan–Dec.

%B specifies the long-month descriptor. The range for the long-month descriptor is January–December.

%c specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.

d specifies the day descriptor and uses 0 padding. The range for the day descriptor is 01–31.

e specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.

%F specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.

%H specifies the hour descriptor that is based on a 24-hour clock. The range for the hour descriptor is 00–23.

%I specifies the hour descriptor that is based on a 12-hour clock. The range for the hour descriptor is 01–12.

%j specifies the day-of-year descriptor as a decimal number and uses a leading zero. The range for the day-of-year descriptor is 1–366.

%m specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.

%M specifies the minute descriptor. The range for the minute descriptor is 00–59.

%O specifies the month descriptor and uses blank padding. The range for the month descriptor is 1–12.

%p specifies a.m. or p.m. descriptor.

%S specifies the second descriptor. The range for the second descriptor is 00–59.

%U specifies the weekday descriptor as a number in the range of 1–7 that represents Monday–Sunday.
%U specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value and uses the number-of-week value within the year (Sunday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53. A leading zero and a maximum value of 53 is used.

%V specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value. The number-of-week value is represented as a decimal number in the range 01–53. A leading zero and a maximum value of 53 is used. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year.

%w specifies the weekday descriptor as a number in the range of 0–6 that represents Sunday–Saturday.

%W specifies the week-number-of-year descriptor by calculating the descriptor value as SAS date value using the number of week within the year (Monday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range of 0–53. A leading zero and a maximum value of 53 are used.

%y specifies the year (2-digit) descriptor. The range for the year descriptor is 00–99.

%Y specifies the year (4-digit) descriptor. The range for the year descriptor is 1970–2069.

Details
The NLDATM function converts the SAS datetime value to the datetime value of the specified locale by using the datetime descriptors.

Example
The following example shows a time (a.m or p.m.) that is created from a SAS datetime value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>options locale=English;</td>
<td>12PM</td>
</tr>
<tr>
<td>time_ampm=nldatm('24Feb2003:12:39:43' dt,'%I%p');</td>
<td></td>
</tr>
<tr>
<td>put time_ampm;</td>
<td></td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
options locale=German; | 12nachm

time_ampm=nldatm('24Feb2003:12:39:43'dt,'%I%p');

put time_ampm;

---

See Also

**Format:**

- "NLDATMw. Format" on page 176

---

**NLTIME Function**

Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.

**Category:** Date and Time

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

**Syntax**

```
NLTIME(time|datetime,descriptor,startpos)
```

**Required Arguments**

- **time**
  - specifies a SAS time value.

- **datetime**
  - specifies a SAS datetime value.

- **descriptor**
  - is a variable, or expression, that specifies the value of a descriptor. You can enter the following descriptors in uppercase or lowercase:
    - `#`
      - removes the leading zero from the result.
    - `%%`
      - specifies the % character.
    - `%H`
      - specifies the hour descriptor that is based on a 24-hour clock. The range for the hour descriptor is 00–23.
    - `%I`
      - specifies the hour descriptor that is based on a 12-hour clock. The range for the hour descriptor is 01–12.
%M
    specifies the minute modifier. The range for the minute descriptor is 00–59.

%P
    specifies the a.m. or p.m. descriptor.

%S
    specifies the second descriptor. The range for the second descriptor is 00–59.

startpos
    is an integer that specifies the position at which the search should start and that specifies the direction of the search.

Details

The NLTIME function converts a SAS time or datetime value to the time value of the specified locale by using the time descriptors.

Example

The following example shows an a.m. or p.m. time that is created from a SAS time.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English; time_ampm=nltime('12:39:43't,'%i%p'); put time_ampm;</td>
<td>12PM</td>
</tr>
<tr>
<td>options locale=German; time_ampm=nltime('12:39:43't,'%i%p'); put time_ampm;</td>
<td>12nachm</td>
</tr>
</tbody>
</table>

See Also

Format:
  - “NLTIMEw. Format” on page 295

SASMSG Function

Specifies a message from a data set. The returned message is based on the current locale and a specified key.

Category: Locale

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.
Syntax

SASMSG (BASENAME, "KEY", "QUOTE|DQUOTE|NOQUOTE">
<, "substitution 1", ..., "substitution 7">)

Required Arguments

BASENAME
  the name of the data set where the message is located.

KEY
  the message key.

Note: If you specify an invalid key name, then the key name is returned.

QUOTE|DQUOTE|NOQUOTE
  specifies the type of quotation marks that are added to the message text and
  substitution strings.

Default: DQUOTE

substitution
  string substitutions. The maximum string substitutions is 7.

Details

The SAS message data set must be a 7-bit ASCII data set. Any character that
cannot be represented in the 7-bit ASCII encoding is represented in the Unicode
escape format of '\uxxxx', where 'xxxx' is the base 10 numeric representation of the
Unicode value of the character.

The data set used by the SASMSG function must have been created specifically for
use with this function. The data set must contain the following variables:

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>locale</td>
<td>char</td>
<td>5</td>
<td>language of the message</td>
</tr>
<tr>
<td>2</td>
<td>key</td>
<td>char</td>
<td>60</td>
<td>key to identify the message</td>
</tr>
<tr>
<td>3</td>
<td>lineno</td>
<td>num</td>
<td>5</td>
<td>line # of the message in reverse order</td>
</tr>
<tr>
<td>4</td>
<td>text</td>
<td>text</td>
<td>1,200</td>
<td>text of the message</td>
</tr>
</tbody>
</table>

The data set must be sorted on the following variables: locale, key, and lineno. The
variable lineno must be in descending order. A composite index on locale and key
must be defined. Here is a sample program to sort and create an indexed data set:
%let basename=MyProduct;
proc sort data=t.&basename;
by locale key descending lineno;
run;
proc datasets lib=t
mctype=data;
modify &basename;
index create indx=(LOCALE KEY);
run;
quit;

The returned message is based on the LOCALE system option. The LOCALE option
is represented by ll_RR where ll represents the two-letter language code and RR
represents the two-letter region code. If a match is not found, then the function
searches for a match with the language only. If the pair locale and key are still not
found, then the function defaults to the English language (en). If the key does not
exist for English (en), then the key name is returned.

You can alter formatting. You can use string substitution by using the format code
%$. You can change the order of substitution. In some cases, translation of a
message to a language other than English might require changing the order of
substitutions. You can change the order by placing an argument number
specification, #nn, within a format string, where nn is the number of the argument in
the substitution list. The following example demonstrates the order:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
</table>
| msg = sasmsg
{"nls.mymsg","IN_CD_LOG",  
"noquote","cat","dog"};
IN_CD_LOGINFO = My %#1s. Your %#2s | msg= My cat. Your dog. |
| IN_CD_LOGINFO = My %#2s. Your %#1s | msg= My dog. Your cat. |

The SASMSG function can be used in the open code macro with the %SYSFUNC
macro function.

Arguments that are passed to a function called by the %SYSFUNC macro must not
be in quotation marks. Arguments passed to the SASMSG function outside of
%SYSFUNC must be quoted.

When the SASMSG function is used with the %SYSFUNC macro function, the
returned string is wrapped with the %NRBQUOTE function.

Examples:

Example 1
The following example demonstrates the formatting feature of SASMSG:

%macro demo_sasmsg;
    data _null_
    msg = sasmsg("nls.mymsg","IN_APW_SAVE_OK","noquote");
    put msg=;
    run;
%mend demo_sasmsg;
### SAS Statements

<table>
<thead>
<tr>
<th>Option Setting</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale = en_US; %demo_sasmsg;</td>
<td>msg=The Access Control key was successfully saved.</td>
</tr>
<tr>
<td>options locale = es_ES; %demo_sasmsg;</td>
<td>msg=La clave de control de acceso se ha guardado.</td>
</tr>
<tr>
<td>options locale = french_France; %demo_sasmsg;</td>
<td>msg=La clé de contrôle d'accès a bien été enregistrée.</td>
</tr>
</tbody>
</table>

### Example 2

The following example demonstrates the open macro feature:

```sas
%MACRO PRT(loc,tb,key);
  option locale=&loc;
  %PUT %SYSFUNC(SASMSG(&tb,&key) );
%MEND PRT;
```

<table>
<thead>
<tr>
<th>SAS Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>%PRT(en_US,&amp;TABLEID,IN_EDIT)</td>
<td>&quot;Edit&quot;</td>
</tr>
<tr>
<td>%PRT(es_ES,&amp;TABLEID,IN_EDIT)</td>
<td>&quot;Editar&quot;</td>
</tr>
<tr>
<td>%PRT(fr_FR,&amp;TABLEID,IN_EDIT)</td>
<td>&quot;Modifier&quot;</td>
</tr>
</tbody>
</table>

### SASMSGL Function

Specifies a message from a data set. The message is based on a specified locale value and a specified key value.

**Category:** Locale  
**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

**Syntax**

```sas
SASMSGL("BASENAME", "KEY", "LOCALE", , , , "Q"|"D"|"N">,<,"substitution 1", ..., "substitution 6">>)
```

**Required Arguments**

**BASENAME**
the name of the data set where the message is located.
**KEY**
the message key.

Note: If you specify an invalid key name, then the key name is returned.

**LOCALE**
the posix locale value (Il_RR).

**QUOTE|DQUOTE|NOQUOTE**
specifies the type of quotation marks that are added to the message text and substitution strings.

Default  **DQUOTE**

**substitution**
string substitutions. The maximum string substitutions is 6.

Details
The SAS message data set must be a 7-bit ASCII data set. Any character that cannot be represented in the 7-bit ASCII encoding is represented in the Unicode escape format of '\uxxxx', where the xxxx is the base 10 numeric representation of the Unicode value of the character.

The data set used by SASMSGL function must have been created specifically for use with this function. The data set must contain the following variables:

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>locale</td>
<td>char</td>
<td>5</td>
<td>language of the message</td>
</tr>
<tr>
<td>2</td>
<td>key</td>
<td>char</td>
<td>60</td>
<td>key to identify the message</td>
</tr>
<tr>
<td>3</td>
<td>lineno</td>
<td>num</td>
<td>5</td>
<td>line number of the message in reverse order</td>
</tr>
<tr>
<td>4</td>
<td>text</td>
<td>text</td>
<td>1200</td>
<td>text of the message</td>
</tr>
</tbody>
</table>

The data set must be sorted on the following variables: **locale**, **key**, and **lineno**. The variable **lineno** must be in descending order. A composite index on locale and key must be defined. Here is a sample program to sort and create an indexed data set:

```sas
%let basename=MyProduct;
proc sort data=t.&basename;
  by locale key descending lineno;
run;

proc datasets lib=t
  memtype=data;
```
modify &basename;
index create indx=(LOCALE KEY);
run;
quit;

The returned message is based on the LOCALE system option. The LOCALE option is represented by ll_RR where ll represents the two-letter language code and RR represents the two-letter region code. If a match is not found, then the function searches for a match with the language only. If the pair locale and key are still not found, then the function defaults to the English language (en). If the key does not exist for English (en), then the key name is returned.

You can alter formatting. You can use string substitution by using the format code %s. You can change the order of substitution. In some cases, translation of a message to a language other than English might require changing the order of substitutions. You can change the order by placing an argument number specification, #nn, within a format string, where nn is the number of the argument in the substitution list. The following example demonstrates changing the order:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_CD_LOGINFO = My %%1s. Your %%2s</td>
<td>IN_CD_LOGINFO = My %%2s. Your %%1s</td>
</tr>
<tr>
<td></td>
<td>msg= My dog. Your cat.</td>
</tr>
</tbody>
</table>

The SASMSGL function can be used in the open code macro with the %SYSFUNC macro function.

Arguments that are passed to a function called by the %SYSFUNC macro must not be in quotation marks. Arguments passed to the SASMSGL function outside of %SYSFUNC must be quoted.

When the SASMSGL function is used with the %SYSFUNC macro function, the returned string is wrapped with the %NRBQUOTE function.

Examples:

Example 1

The following example demonstrates the formatting feature of SASMSGL:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>sasmsgl(“nls.mymag”, “IN_APW_SAVE_OK”, “en_US”, “Access key was successfully saved.”)</td>
<td>sasmsgl(“nls.mymag”, “IN_APW_SAVE_OK”, “es_ES”, “clave del acceso se ha guardado.”)</td>
</tr>
<tr>
<td>sasmsgl(“nls.mymag”, “IN_APW_SAVE_OK”, “fr_FR”, “Clé de contrôle d’accès a bien été enregistrée.”)</td>
<td>sasmsgl(“nls.mymag”, “IN_APW_SAVE_OK”, “en_US”, “Access key was successfully saved.”)</td>
</tr>
</tbody>
</table>
Example 2

The following example demonstrates the open macro feature:

<table>
<thead>
<tr>
<th>SAS Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, en_US));</td>
<td></td>
</tr>
<tr>
<td>$PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, es_ES));</td>
<td>&quot;Editar&quot;</td>
</tr>
<tr>
<td>$PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, fr));</td>
<td>&quot;Modifier&quot;</td>
</tr>
</tbody>
</table>

SETLOCALE Function

Specifies the locale keys for the current SAS locale.

Category: Locale

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

Setting SAS Locale

SETLOCALE (sas_locale)

Customize single locale elements

SETLOCALE (key, value)

Customize single locale elements

SETLOCALE (category_name, sas_locale)

Required Arguments

sas_locale

specifies a SAS locale name by using the SAS name or the posix name. You can also specify the locale alias.

key

specifies a SAS locale element key. See the list of element keys in the Details section.

value

specifies a value for the locale element.

category_name

specifies the category name:

- LC_TIME
- LC_MONETARY
- LC_NUMERIC
Details

You can modify the following locale elements. The value of key must be less than the value of max length. You can specify the following values for type:

- 0  String.
- 1  Unsigned integer. You must use double quotation marks.

<table>
<thead>
<tr>
<th>Locale Element Key</th>
<th>Max Length</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATESTYLE</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PAPERSIZE</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTITLE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTEXT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMFONT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SORTSEQ</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>MESSAGES</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATETIME</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_TIME</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_NUMERIC</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_PERCENT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SANSSERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_CURSIVE</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_FANTASY</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------</td>
<td>------</td>
<td>--------------</td>
</tr>
<tr>
<td>FONT_MONOSPACED</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BRUSH</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SWISS</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ITALIC</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DATE_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_AMPM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_WEEK_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_WEEK_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME_AMPM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_WEEK_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_WEEK_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYMM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>DATE_YYMM_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>ABMON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>DAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>AM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>PM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>INT_CURRENCY_SYMBOL</td>
<td>3</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>CURRENCY_SYMBOL</td>
<td>32</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_DECIMAL_POINT</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_THOUSANDS_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_NEGATIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>MON_INT_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>DECIMAL_POINT</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>THOUSANDS_SEP</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>NEGATIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
</tbody>
</table>
### Examples:

#### Example 1

In the following locale example, the SETLOCALE function specifies the locale Japanese (jp_JP). The SETLOCALE function returns the previous locale. In this example, the previous locale was English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x=setlocale('ja_JP');</td>
<td>x=English_UnitedStates</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

#### Example 2

In the following example, the SETLOCALE function returns the locale name where the element values are being changed:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x = setlocale(&quot;LC_MONETARY&quot;, 'zh_CN');</td>
<td>x=Japanese_Japan</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

#### Example 3

In the following example, the SETLOCALE function changes the value of the specified key, DATE_YEAR_FORMAT:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x=setlocale('DATE_YEAR_FORMAT', '¥%Y');</td>
<td>x=¥%Y</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
SORTKEY Function

Creates a linguistic sort key.

Category: Locale

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\[ \text{sortKey}(\text{string}, \langle \text{locale}, \text{strength}, \text{case_order}, \text{numeric_order} \rangle) \]

Required Arguments

\textbf{string}
character expression

\textbf{locale}
specifies the locale name in the form of a POSIX name (ja_JP). See Table 21.1 on page 811 for a list of locale names and POSIX values.

\textbf{strength}
The value of strength is related to the collation level. There are five collation-level values. The following table provides information about the five levels. The default value for strength is related to the locale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Type of Collation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY or P</td>
<td>PRIMARY specifies differences between base characters (for example, &quot;a&quot; &lt; &quot;b&quot;).</td>
<td>It is the strongest difference. For example, dictionaries are divided into different sections by base character.</td>
</tr>
<tr>
<td>SECONDARY or S</td>
<td>Accents in the characters are considered secondary differences (for example, &quot;as&quot; &lt; &quot;às&quot; &lt; &quot;at&quot;).</td>
<td>Other differences between letters can also be considered secondary differences, depending on the language. A secondary difference is ignored when there is a primary difference anywhere in the strings.</td>
</tr>
<tr>
<td>TERTIARY or T</td>
<td>Upper and lowercase differences in characters are distinguished at the tertiary level (for example, &quot;ao&quot; &lt; &quot;Ao&quot; &lt; &quot;aö&quot;).</td>
<td>An example is the difference between large and small Kana. A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings.</td>
</tr>
</tbody>
</table>
When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, "ab" < "a-b" < "aB"). This difference is ignored when there is a primary, secondary, or tertiary difference. The quaternary level should be used if ignoring punctuation is required or when processing Japanese text.

When all other levels are equal, the identical level is used as a tiebreaker. The Unicode code point values of the NFD form of each string are compared at this level, just in case there is no difference at levels 1-4. For example, only Hebrew cantillation marks are distinguished at this level. This level should be used sparingly, as only code point values differences between two strings is an extremely rare occurrence.

case order
sorts uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL. The following table provides the values and information for the case order argument.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER or U</td>
<td>Sorts uppercase letters first, then the lowercase letters.</td>
</tr>
<tr>
<td>LOWER or L</td>
<td>Sorts lowercase letters first, then the uppercase letters.</td>
</tr>
</tbody>
</table>

numeric order
orders numbers by the numeric value instead of the number's characters.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC or N</td>
<td>Order numbers (integers) by the numeric value. For example, &quot;8 Main St.&quot; would sort before &quot;45 Main St.&quot;.</td>
</tr>
</tbody>
</table>

collation order
There are two types of collation values: Phonebook and Traditional. If you do not select a collation value, then the user's locale-default collation is selected. The following table provides more information.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHONEBOOK or P</td>
<td>specifies a phonebook style ordering of characters. Select PHONEBOOK only with the German language.</td>
</tr>
<tr>
<td>TRADITIONAL or T</td>
<td>specifies a traditional style ordering of characters. Select TRADITIONAL only with the Spanish language.</td>
</tr>
</tbody>
</table>
Details

The SORTKEY function creates a linguistic sort key for data. You must enter at least one argument. If the length of the variable that receives the key is not large enough, the data truncates, and a warning is displayed.

locale
Locale values use the POSIX name (ll_RR). LL represents the two-letter language code, and RR represents the two-letter region code. For example, en_US is the POSIX name for English, United States. en represents the English language, and US represents the United States. If a locale value is not specified, then the session locale is used.

strength
The strength argument determines whether accents or case affect collating or matching text. If no value is specified for strength, then the locale determines the value. The following values can be specified for strength.

- PRIMARY
  This value includes base letters. An example is the letters, A, a, and Å are all processed the same.

- SECONDARY
  This value processes data the same as PRIMARY, and accents are processed. The letters A and a are processed equally, and Å is processed as an accented character.

- TERTIARY
  This value processes data the same as SECONDARY, and the character's case is processed. For example, A, a, and Å are all processed differently.

- QUATERNARY
  This value processes data the same as TERTIARY, and punctuation is processed.

- IDENTICAL
  This value process data the same as QUATERNARY, and code point is processed.

case order
specifies to sort data using uppercase or lowercase letter. The following table shows examples of specifying the UPPER value or the LOWER value.

<table>
<thead>
<tr>
<th>UPPER</th>
<th>LOWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aztec</td>
<td>aztec</td>
</tr>
<tr>
<td>aztec</td>
<td>Aztec</td>
</tr>
<tr>
<td>Mars</td>
<td>mars</td>
</tr>
<tr>
<td>mars</td>
<td>Mars</td>
</tr>
</tbody>
</table>

collation order
The collation order value PHONEBOOK is ignored unless the locale is a German language.

The collation order value TRADITIONAL is ignored unless the locale is a Spanish language.
A warning message is displayed for other locales.

## TZONEID Function

Returns the current time zone ID.

**Category:** Date and Time  
**Alias:** TZID  
**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

### Syntax

**TZONEID**<time-zone-id>

### Optional Argument

**time-zone-id**  
specifies a region/area value that is defined by SAS. When you specify a zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

### Details

The TZONEID function returns a blank value if the TIMEZONE= option is blank or a user-defined time zone is specified.

The TZONEID function validates the timezone ID. If you specify the timezone ID, the function returns the timezone ID if it is valid or returns a blank value if the ID is invalid.

### Example

In the first example, the TIMEZONE option is set to JST. In the second example, TIMEZONE is set to a blank value. In the third example TIMEZONE is set to user-specified time zone. In the fourth example a valid timezoneid and an invalid timezoneid is displayed.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options timezone=jst;</code></td>
<td></td>
</tr>
<tr>
<td><code>data _null_ ;</code></td>
<td></td>
</tr>
<tr>
<td><code>   tzid=tzoneid() ;</code></td>
<td></td>
</tr>
<tr>
<td><code>   put tzid=;</code></td>
<td>tzid=ASIA/TOKYO</td>
</tr>
<tr>
<td><code>run;</code></td>
<td></td>
</tr>
</tbody>
</table>
Statements | Results
---|---
```options timezone='';
data _null_;   
    tzid=tzoneid();   
    put tzid=;  
run;```

```options timezone='xxx-12';   
/* user defined timezone */
data _null_;   
    tzid=tzoneid();   
    put tzid=;  
run;```

```data null;   
    name_valid=tzoneid('asia/tokyo');   
    name_invalid=tzoneid('Milky Way');   
    put name_valid =;   
    put name_invalid=;  
run;```

TZONENAME Function

Returns the current standard or daylight savings time, time zone name.

Category: Date and Time

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

```
TZONENAME()

TZONENAME<time-zone-id,datetime>
```

Optional Arguments

`time-zone-id`

specifies a `region/area` value that is defined by SAS. When you specify a zone ID, the time zone that SAS uses is determined by time zone name and daylight savings time rules.

See For a list of the time zone IDs, see Appendix 3, “Time Zone IDs and Time Zone Names,” on page 953.

`datetime`

specifies a SAS datetime value.
Details

The `TZONENAME` function returns a blank value if the `TIMEZONE=` option and the time-zone-id argument are blank.

The `TZONENAME` function returns the timezone name based on the specified timezone and datetime. If the SAS datetime is not specified, then the current date is used. If time-zone-id is not specified, then the timezone ID that is specified with the `TIMEZONE=` option is used.

Example

In the first example, the `TIMEZONE` option is set to a blank value. In the second example, `TIMEZONE` is set to timezone name, JST. In the third example, `TIMEZONE` is set to a user-specified time zone. In the fourth example, `TIMEZONE` is set to a time zone ID.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options tz='';</td>
<td>tzname=</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzone=tzonename() ;</td>
<td></td>
</tr>
<tr>
<td>put tzone =;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options tz='jst';</td>
<td>tzname=JST</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzone=tzonename();</td>
<td></td>
</tr>
<tr>
<td>put tzone =;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options tz='xxx-12';</td>
<td>tzname=XXX</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzone=tzonename();</td>
<td></td>
</tr>
<tr>
<td>put tzone =;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options tz='America/Chicago';</td>
<td>tzname=CDT</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzone=tzonename('01SEP2014:01:01:01'dt);</td>
<td></td>
</tr>
<tr>
<td>put tzone =;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**TZONEOFF Function**

Returns the user time zone offset.

**Category:** Date and Time

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see *Internationalization Compatibility on page 343.*
Syntax

TZONEOFF()

TZONEOFF(<time-zone-id, datetime>)

Optional Arguments

time-zone-id
 specifies a region/area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by time zone name and daylight savings time rules.

See For a list of time zone IDs, see Appendix 3, “Time Zone IDs and Time Zone Names,” on page 953.

datetime
 specifies a SAS datetime value.

Details

If no arguments are specified, the TZONEOFF function returns the time zone offset for the specified TIMEZONE option. The TZONEOFF (time-zone-id) function with the time zone ID argument returns the time zone offset for the specified time zone ID. The TZONEOFF function with the time zone ID name returns the time zone offset for the specified time zone name. We recommend that you use the time zone ID, since it is not locale dependent.

If SASDTM is not provided, TZONEOFF returns the current timezone offset. If SASDTM is provided, it returns the offset to get the local time for specified time value.

Example

The first example has no argument, so the TZONEOFF function returns an offset for the current SAS session. The second example returns an offset based on a specific time zone ID. The third example returns an offset based on a specific time zone ID and a specific date and time. The fourth example returns an offset based on the Time Zone option and a specific date.

If the SAS datetime is not specified, then the TZONEOFF function returns the current timezone offset. If the SAS datetime is specified, then the function returns the offset to provide the local time for the specified time value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option TIMEZONE='AUSTRALIA/MELBOURNE'; %PUT %SYSFUNC(TZONEOFF());</td>
<td>39600</td>
</tr>
<tr>
<td>option TIMEZONE='AUSTRALIA/MELBOURNE'; %PUT %SYSFUNC(TZONEOFF('EUROPE/ROME'));</td>
<td>3600</td>
</tr>
</tbody>
</table>
Statements

```sas
options TIMEZONE='EUROPE/MOSCOW';
data _null_;  
  dt1='05DEC2012:08:17:52'dt  ;
  dt2='05JUN2012:08:17:52'dt  ;
  offset1= TZONEOFF('EUROPE/MOSCOW', dt1) ;
  offset2= TZONEOFF('EUROPE/MOSCOW', dt2) ;
  put offset1= / offset2= ;
run ;
```

Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>offset1=14400</td>
<td>offset2=14400</td>
</tr>
</tbody>
</table>

```sas
options TIMEZONE='EUROPE/MOSCOW';
data _null_;  
  dt1='05DEC2012:08:17:52'dt  ;
  dt2='05JUN2012:08:17:52'dt  ;
  offset1= TZONEOFF(dt1) ;
  offset2= TZONEOFF(dt2) ;
  put offset1= / offset2= ;
run ;
```

Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>offset1=14400</td>
<td>offset2=14400</td>
</tr>
</tbody>
</table>

---

**TZONES2U Function**

Converts a SAS date time value to a UTC date time value.

**Category:** Date and Time

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see *Internationalization Compatibility* on page 343.

**Syntax**

```
TZONES2U <datetime, time-zone-id>
```

**Required Argument**

`datetime`

specifies a SAS datetime value.

**Optional Argument**

`time-zone-id`

specifies a region/area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by time zone name and daylight savings time rules.

**See** For a list of time zone IDs, see *Appendix 3, “Time Zone IDs and Time Zone Names,”* on page 953
Details

The TZONES2U() function returns UTC-based time for the specified TIMEZONE. The TZONES2U(time-zone-id) function with the time zone ID argument returns UTC-based time for the specified time zone ID. If the time zone name is not valid for the current locale, you receive an error. If time-zone-id is not specified, then the timezone ID that is specified with the TIMEZONE= option is used.

Example

The following example converts a SAS date time into UTC time.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=ja_JP TZ='JST' ;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>data <em>null</em> ; dt='05Nov2012:08:17:52'dt ; utc1 = tzones2u(dt) ;</td>
<td>utc1=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>utc2 = tzones2u(dt,'ASIA/TOKYO') ;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>utc3 = tzones2u(dt,'JST') ;</td>
<td>utc3=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>put dt= /utc1= is8601dz. //;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>put dt= /utc2= is8601dz. // ;</td>
<td>utc2=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>put dt= /utc3= is8601dz. // ;</td>
<td>utc3=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>run ;</td>
<td></td>
</tr>
</tbody>
</table>
Details

If the time-zone-id is not specified, then the TIMEZONE ID that is specified by the TIMEZONE= option is used. If the daylight savings time does not exist for the specified time zone, then the function returns a blank name.

Example

The following example specifies the America/Chicago time zone.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone='America/Chicago'; dstname=CDT</td>
<td></td>
</tr>
<tr>
<td>data null; dstname=tzonedstname(); put dstname; run;</td>
<td></td>
</tr>
<tr>
<td>dstname=CDT</td>
<td></td>
</tr>
</tbody>
</table>

TZONEDSTOFF Function

Returns the time zone offset value for the specified daylight savings time.

Category: Date and Time

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

TZONEDSTOFF()
TZONEDSTOFF<time-zone-id>

Optional Argument

time-zone-id

specifies a region/area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details

If the time-zone-id is not specified, then the TIMEZONE ID that is specified by the TIMEZONE= option is used. If the daylight savings time does not exist for the specified time zone, then the function returns a blank name.

Example

The following example specifies the America/Chicago time zone.
options timezone='America/Chicago';
data null;
  dstoff=tzonedstoff();
  put dstoff;
run;
dstoff=18000

TZONESTTNAME Function

Returns a standard time zone name.

Category: Date and Time

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

TZONESTTNAME()
TZONESTTNAME<time-zone-id>

Optional Argument

time-zone-id
  specifies a region or area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details

If the TZID is not specified, then the TIMEZONE ID that is specified by the TIMEZONE= option is used.

Example

The following example specifies the Asia/Osaka time zone.

options timezone='Asia/Osaka';
data null;
  name=tzonesttname();
  put name;
run;
JST
TZONESTTOFF Function

Returns the time zone offset value for the specified standard time.

Category: Date and Time

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

TZONESTTOFF()
TZONESTTOFF<time-zone id>

Optional Argument

time-zone-id
specifies a region or area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details

If the time-zone-id is not specified, then the TIMEZONE ID that is specified by the TIMEZONE= option is used. If standard time is not specified, then the function returns a blank name.

Example

The following example specifies the Asia/Osaka time zone.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| options timezone='Asia/Osaka';
data null;
   name=tzonesttoff();
   put name;
run;       | 32400   |

TZONEU2S Function

Converts a UTC date time value to a SAS date time value.

Category: Date and Time
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

TZONEU2S <UTC date time value, time-zone id>

Required Argument

UTC date time value
specifies a Coordinated Universal Time (UTC) datetime value

Optional Argument

time-zone-id
specifies a region/area value that is defined by SAS. When you specify a zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details

The TZONEU2S(datetime) function returns the SAS datetime for a UTC time for the specified TIMEZONE option. The TZONEU2S(datetime, time-zone-id) function with the time zone ID argument, returns the SAS datetime for the UTC time for the specified time zone ID.

Example

The following example converts a UTC date time to three specific SAS date time values.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=fr_FR TZ='AMERICA/DENVER'; data <em>null</em>; utc_date = '2012-09-02T02:34:56+00:00'; udt = input(utc_date,is8601dz.); sdt1 = tzoneu2s(udt); sdt2 = tzoneu2s(udt,'EUROPE/AMSTERDAM'); sdt3 = tzoneu2s(udt,'CET'); put sdt1= datetime. / sdt2= datetime. / sdt3= datetime.; run;</td>
<td>sdt1=01SEP12:20:34:56 sdt2=02SEP12:04:34:56 sdt3=02SEP12:03:34:56</td>
</tr>
</tbody>
</table>

TRANTAB Function

Transcodes data by using the specified translation table.

Category: Character
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

**TRANTAB**(string, trantab_name)

Required Arguments

- **string**: input data that is transcoded.
- **trantab_name**: translation table.

Details

The TRANTAB function transcodes a data string by using a translation table to remap the characters from one internal representation to another. The encoding of the data in the input string must match the encoding of table 1 in the translation table. The TRANTAB function remaps the data from the encoding using table 1.

Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= and ENCODING= system options as an improvement on direct use of translation tables. In SAS 9 translation tables are used only for transcoding external files. There is direct transcoding in SAS files, between the session encodings. SAS 9.2 supports the TRANTAB function for backward compatibility.

The LOCALE= system option is preferred in later SAS releases.

**CAUTION**

Only experienced SAS users should use the TRANTAB function.

Example

The following example uses a translation table that transcodes data that is encoded in Latin2 to an uppercase Latin2 encoding:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>teststrg=trantab('testing','lat2_ucs'); put teststrg;</td>
<td>TESTING</td>
</tr>
</tbody>
</table>

See Also

**Procedures:**

- Chapter 20, “TRANTAB Procedure,” on page 783
UNICODE Function

Converts Unicode characters to the current SAS session encoding.

Category: Character

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\[ STR=UNICODE(<instr> ,<Unicode type> ) \]

Required Argument

\( str \)

Data string that has been converted to the current SAS session encoding.

Optional Arguments

\( instr \)

input data string.

\( Unicode type \)

Unicode character formats

- **ESC**: Unicode Escape (for example, \u0042 ). ESC is the default format.
- **NCR**: Numeric Character Representation (for example, &#22823 or &#177 ; )
- **PAREN**: Unicode Parenthesis Escape (for example, <u0061>)
- **UTF8**: UTF8 encoding.
- **UTF16**: UTF16 encoding with big endian. UCS2 is an alias.
- **UTF16B**: UTF16 encoding with big endian. UCS2B is an alias.
- **UTF16L**: UTF16 encoding with little endian. UCS2L is an alias.
- **UTF32**: UTF32 encoding with big endian. UCS4 is an alias.
- **UTF32B**: UTF32 encoding with big endian. UCS4B is an alias.
- **UTF32L**: UTF32 encoding with little endian. UCS4L is an alias.

Details

This function reads Unicode characters and converts them to the current SAS session encoding.

The Unicode function supports the Unicode supplementary characters.
Example

The following example demonstrates the functionality of the UNICODE function:

/* Run this program using any CJK LOCALE */
data _null_
  str1=unicode("\u0041\u0042\u0043");put str1=; /* default */
  str2=unicode("\u0041\u0042\u0043", "esc");put str2=; /* ESC - Unicode Escape */
  str3=unicode("\&#177;", "ncr");put str3=; /* NCR - Numeric Character Representation */
  str4=unicode("\&#22823;", "ncr");put str4=; /* NCR - Numeric Character Representation */
  str5=unicode('<u0061><u0062>', 'paren');put str5=; /* PAREN - Unicode Parenthesis Escape */
  str6=unicode('2759'x, 'ucs2');put str6=; /* UCS2 - UCS2 encoding */
  str7=unicode('5927'x, 'ucs2b');put str7=; /* UCS2B - UCS2 encoding with big endian */
  str8=unicode('2759'x, 'ucs2l');put str8=; /* UCS2L - UCS2 encoding with little endian */
  str9=unicode('27590000'x, 'ucs4');put str9=; /* UCS4 - UCS4 encoding */
  str10=unicode('00005927'x, 'ucs4b');put str10=; /* UCS4B - UCS4 encoding with big endian */
  str11=unicode('27590000'x, 'ucs4l');put str11=; /* UCS4L - UCS4 encoding with little endian */
  str12=unicode('E5A4A7'x, 'utf8');put str12=; /* UTF8 - UTF8 encoding */
  str13=unicode('2759'x, 'utf16');put str13=; /* UTF16 - UTF16 encoding */
  str14=unicode('5927'x, 'utf16b');put str14=; /* UTF16B - UTF16 encoding with big endian */
  str15=unicode('2759'x, 'utf16l');put str15=; /* UTF16L - UTF16 encoding with little endian */
  str16=unicode("\u00020bb7", "esc");put str16=; /* ESC - Unicode Escape for Supplementary Character */
  str17=unicode("\&#134071;", "ncr");put str17=; /* NCR - Numeric Character Representation for Supplementary Character */
  str18=unicode('<u00020BB7>', 'paren');put str18=; /* PAREN - Unicode Parenthesis Escape for Supplementary Character */
run;

Here are the results from the UNICODE function example:

<table>
<thead>
<tr>
<th>str1</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>str2</td>
<td>ABC</td>
</tr>
<tr>
<td>str3</td>
<td>±</td>
</tr>
<tr>
<td>str4</td>
<td>大</td>
</tr>
<tr>
<td>str5</td>
<td>大</td>
</tr>
<tr>
<td>str6</td>
<td>大</td>
</tr>
<tr>
<td>str7</td>
<td>大</td>
</tr>
<tr>
<td>str8</td>
<td>大</td>
</tr>
<tr>
<td>str9</td>
<td>大</td>
</tr>
<tr>
<td>str10</td>
<td>大</td>
</tr>
<tr>
<td>str11</td>
<td>大</td>
</tr>
<tr>
<td>str12</td>
<td>大</td>
</tr>
<tr>
<td>str13</td>
<td>大</td>
</tr>
<tr>
<td>str14</td>
<td>大</td>
</tr>
<tr>
<td>str15</td>
<td>大</td>
</tr>
<tr>
<td>str16</td>
<td>大</td>
</tr>
<tr>
<td>str17</td>
<td>大</td>
</tr>
<tr>
<td>str18</td>
<td>大</td>
</tr>
</tbody>
</table>
UNICODEC Function

Converts characters in the current SAS session encoding to Unicode characters.

Category: Character

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

\[
\text{STR} = \text{UNICODEC}(\langle \text{instr} \rangle, (\langle \text{Unicode type} \rangle))
\]

Required Argument

\textit{str}

data string that has been converted to Unicode encoding.

Optional Arguments

\textit{instr}

input data string.

\textit{Unicode type}

Unicode character formats

- ESC: Unicode Escape (for example, \u0042) ESC is the default format.
- NCR: Numeric Character Representation (for example, &\#22823 or &\#177 ;)
- PAREN: Unicode Parenthesis Escape (for example, <u0061>)
- UTF8: UTF8 encoding.
- UTF16: UTF16 encoding with big endian. UCS2 is an alias.
- UTF16B: UTF16 encoding with big endian. UCS2B is an alias.
- UTF16L: UTF16 encoding with little endian. UCS2L is an alias.
- UTF32: UTF32 encoding with big endian. UCS4 is an alias.
- UTF32B: UTF32 encoding with big endian. UCS4B is an alias.
- UTF32L: UTF32L encoding with big endian. UCS4L is an alias.

Details

This function reads characters that are in the current SAS session encoding and converts them to Unicode encoding.
Examples:

Example 1

The following example demonstrates the functionality of the UNICODEC function:

```sas
data _null_;  
str1=unicodec("ABC", 'utf8');  
put str1= $hex12.;  
str2=unicodec("ABCé", 'utf8');  
put str2= $hex12.;  
run;
```

SAS writes the following output to the log:

```
str1=414243202020  
str2=414243C3A920
```

Example 2

Here is an example of UNICODEC with escape characters:

```sas
data _null_;  
txt   = 'ABC&ABC';  
code1 = unicodec(txt,'ESC');  
code2 = unicodec(txt,'NCR');  
code3 = unicodec(txt,'PAREN');  
put code1= / code2= / code3=;  
run;
```

SAS writes the following output to the log:

```
code1=ABC\uFF21\uFF22\uFF23  
code2=ABC&ABC  
code3=<u0041><u00412><u0043><u0026><uFF21><uFF22><uFF23>
```

UNICODELEVEN Function

Specifies the length of the character unit for the Unicode data.

**Category:** Character

**Restriction:**

This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see [Internationalization Compatibility](#) on page 343.

**Syntax**

```
UNICODELEVEN()
```
Details

The UNICODELEN function specifies the length of the character unit for the UNICODE data.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1=unicodelen(&quot;abc \u5927&quot;);</td>
<td>len1=4</td>
</tr>
<tr>
<td>len2=unicodelen(&quot;\u0041\u0042\u0043\u5927&quot;,&quot;esc&quot;);</td>
<td>len2=4</td>
</tr>
<tr>
<td>len3=unicodelen(&quot;&amp;#22823;&quot;,&quot;ncr&quot;);</td>
<td>len3=1</td>
</tr>
<tr>
<td>len4=unicodelen(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;,&quot;paren&quot;);</td>
<td>len4=2</td>
</tr>
</tbody>
</table>

See Also

Functions:

- "UNICODEWIDTH Function" on page 511

UNICODEWIDTH Function

Specifies the length of a display unit for the Unicode data.

Category: Character

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 343.

Syntax

UNICODEWIDTH()
and CJK Compatibility Ideographs Supplement have the value of a display unit 2. Other characters are display unit 1.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1=unicodewidth(&quot;abc &quot;);</td>
<td>len1=5</td>
</tr>
<tr>
<td>len2=unicodewidth(&quot;\u0041\u0042\u0043\u5927&quot;,&quot;esc&quot;);</td>
<td>len2=5</td>
</tr>
<tr>
<td>len3=unicodewidth(&quot;&amp;#22823; &quot;,&quot;ncr&quot;);</td>
<td>len3=2</td>
</tr>
<tr>
<td>len4=unicodewidth(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;,&quot;paren&quot;);</td>
<td>len4=2</td>
</tr>
</tbody>
</table>

See Also

Functions:
- "UNICODELEN Function" on page 510

VARTRANSCODE Function

Returns the transcode attribute of a SAS data set variable.

Category: Variable Information

Syntax

VARTRANSCODE(data-set-id, var-num)

Required Arguments

*data-set-id*

specifies the data set identifier that the OPEN function returns.

*var-num*

specifies the position of the variable in the SAS data set.

Tip  The VARNUM function returns this value.
Details

Transcoding is the process of converting data from one encoding to another. The VARTRANSCODE function returns 0 if the var-num variable does not transcode its value, or 1 if the var-num variable transcodes its value.


Example

The following example shows how to determine whether a character variable is transcoded:

data a;
  attrib x length=$3. transcode=no;
  attrib y length=$3. transcode=yes;
  x='abc';
  y='xyz';
run;
data _null_;  
dsid=open('work.a','i');
nobs=attrn(dsid,"nobs");
nvars=attrn(dsid,"nvars");
do i=1 to nobs;
  xrc=fetch(dsid,1);
  do j=1 to nvars;
    transcode = vartranscode(dsid,j);
    put transcode=;
  end;
end;
run;

SAS writes the following output to the log:

transcode=0
transcode=1

See Also

Functions:
- “ATTRN Function” in SAS Functions and CALL Routines: Reference
- “OPEN Function” in SAS Functions and CALL Routines: Reference
- “VARNUM Function” in SAS Functions and CALL Routines: Reference
- “VTRANSCODE Function” on page 514
- “VTRANSCODEX Function” on page 515
VTRANSCODE Function

Returns a value that indicates whether transcoding is enabled for the specified character variable.

Categories:
- CAS
- Variable Information

Syntax

VTRANSCODE (var)

Required Argument

var

specifies a character variable that is expressed as a scalar or as an array reference.

Restriction

You cannot use an expression as an argument.

Details

The VTRANSCODE function returns 0 if transcoding is off, and 1 if transcoding is on.

By default, all character variables in the DATA step are transcoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

Comparisons

- The VTRANSCODE function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however, evaluates the argument to determine the variable name. The function then returns the transcoding status (on or off) that is associated with that variable name.
- The VTRANSCODE function does not accept an expression as an argument. The VTRANSCODEX function accepts expressions, but the value of the specified expression cannot denote an array reference.
- Related functions return the value of other variable attributes, such as the variable name, type, format, and length. For a list of the variable attributes, see the “Variable Information” functions in SAS Functions and CALL Routines: Reference.

Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
## VTRANSCODEX Function

**Statements**

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrib x transcode = yes;</td>
<td>rc1=0</td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td></td>
</tr>
<tr>
<td>rc1 = vtranscode(y);</td>
<td></td>
</tr>
<tr>
<td>put rc1=;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Functions:**
- “VTRANSCODEX Function” on page 515

**Statements:**
- ATTRIB in

### VTRANSCODEX Function

**Returns a value that indicates whether transcoding is enabled for the specified argument.**

**Categories:** CAS

**Variable Information**

**Syntax**

**VTRANSCODEX (var)**

**Required Argument**

`var` specifies any SAS character expression that evaluates to a character variable name.

**Restriction**

The value of the specified expression cannot denote an array reference.

### Details

The VTRANSCODEX function returns 0 if transcoding is off, and 1 if transcoding is on.

By default, all character variables in the DATA step are transcoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

### Comparisons

- The **VTRANSCODE** function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however,
evaluates the argument to determine the variable name. The function then
returns the transcoding status (on or off) that is associated with that variable
name.

- The VTRANSCODE function does not accept an expression as an argument.
The VTRANSCODEX function accepts expressions, but the value of the
specified expression cannot denote an array reference.

- Related functions return the value of other variable attributes, such as the
variable name, type, format, and length. For a list of the variable attributes, see
the “Variable Information” functions in *SAS Functions and CALL Routines: Reference*.

Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrib x transcode = yes;</td>
<td>rc1=0</td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td>rc1 = vtranscodex('y');</td>
</tr>
<tr>
<td>put rc1=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:
- “VTRANSCODE Function” on page 514

Statements:
- ATTRIB
Dictionary of Informats for NLS

Informats by Category ........................................................................................................................................ 521

Dictionary .................................................................................................................................................. 535
$SCPTDWW. Informat .................................................................................................................................. 535
$SCPTWDw. Informat .................................................................................................................................. 536
EUROW.d Informat ...................................................................................................................................... 537
EUROXw.d Informat ................................................................................................................................... 539
JNENGOw. Informat .................................................................................................................................... 540
$KANJiw. Informat ....................................................................................................................................... 541
$KANJIXw. Informat ..................................................................................................................................... 542
$LOGVSw. Informat ..................................................................................................................................... 543
$LOGVSRe. Informat .................................................................................................................................... 544
MINGUOw. Informat ..................................................................................................................................... 546
NENGOw. Informat ....................................................................................................................................... 547
NLDATWw. Informat ..................................................................................................................................... 548
NLDATWw. Informat ..................................................................................................................................... 549
NLDATMw. Informat ..................................................................................................................................... 550
NLDATMAP Informat ................................................................................................................................... 551
NLDATMMw. Informat .................................................................................................................................. 552
NLMNIAEDw.d Informat ................................................................................................................................ 553
NLMNIAUDw.d Informat ................................................................................................................................ 554
NLMNIBGNw.d Informat ................................................................................................................................ 555
NLMNIBRLw.d Informat ................................................................................................................................ 556
NLMNICADw.d Informat ................................................................................................................................ 557
NLMNICHFw.d Informat ................................................................................................................................ 558
NLMNICNYw.d Informat ................................................................................................................................ 559
NLMNICZKw.d Informat ................................................................................................................................ 560
NLMNIDKKw.d Informat ................................................................................................................................ 561
NLMNIEEKw.d Informat ................................................................................................................................ 562
NLMNIEGPw.d Informat ................................................................................................................................ 563
NLMNIEURw.d Informat ................................................................................................................................ 564
NLMNIGBPw.d Informat ................................................................................................................................ 565
NLMNIHKDw.d Informat ................................................................................................................................ 566
NLMNIHRKw.d Informat ................................................................................................................................ 567
NLMNIHUFw.d Informat ................................................................................................................................ 568
NLMNIIDRw.d Informat ................................................................................................................................ 569
NLMNIILSw.d Informat ................................................................................................................................ 570
NLMNIINRw.d Informat ................................................................................................................................ 571
NLMNIJPYw.d Informat .................................................................................................................................. 572
NLMNIKRWw.d Informat ................................................................................................................................ 573
NLMNILTLw.d Informat .................................................................................................................................. 574
NLMNILVlw.d Informat .................................................................................................................................. 575
NLMNIMOOpw.d Informat ............................................................................................................................. 576
Informats by Category

Some informats run in SAS only, and some informats run in SAS and on the CAS engine. If CAS is specified for the informat category, then the informat runs in SAS and on the CAS server. If CAS is not specified for the informat category, then the informat runs in SAS only. For example, the NLDATE informat runs in SAS and on the CAS server, so CAS is specified as a category. The $UNCR informat runs on SAS only, so CAS is not specified as a category.

Here are categories related to NLS issues:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI Text Handling</td>
<td>Instructs SAS to read bidirectional data values from data variables.</td>
</tr>
<tr>
<td>CAS</td>
<td>Instructs SAS that these informats run on the CAS server.</td>
</tr>
<tr>
<td>Character</td>
<td>Instructs SAS to read character data values into character variables.</td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DBCS</td>
<td>Instructs SAS to manage various Asian languages.</td>
</tr>
<tr>
<td>Date</td>
<td>Instructs SAS to read data values into variables that represent dates.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>Instructs SAS to read data values into variables that represent dates, times, and datetimes.</td>
</tr>
<tr>
<td>Hebrew Text Handling</td>
<td>Instructs SAS to read Hebrew data from data variables.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Instructs SAS to read numeric data values into numeric variables.</td>
</tr>
<tr>
<td>Time</td>
<td>Instructs SAS to read data values into variables that represent times.</td>
</tr>
</tbody>
</table>

The following table provides brief descriptions of the SAS informats. For more detailed descriptions, see the NLS entry for each informat.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI Text Handling</td>
<td>$LOGVSw. Informat (p. 543)</td>
<td>Reads a character string that is in left-to-right logical order, and then converts the character string to visual order.</td>
</tr>
<tr>
<td></td>
<td>$LOGVSRw. Informat (p. 544)</td>
<td>Reads a character string that is in right-to-left logical order, and then converts the character string to visual order.</td>
</tr>
<tr>
<td></td>
<td>$VSLOGw. Informat (p. 662)</td>
<td>Reads a character string that is in visual order, and then converts the character string to left-to-right logical order.</td>
</tr>
<tr>
<td></td>
<td>$VSLOGRw. Informat (p. 663)</td>
<td>Reads a character string that is in visual order, and then converts the character string to right-to-left logical order.</td>
</tr>
<tr>
<td>CAS</td>
<td>NLDATEw. Informat (p. 548)</td>
<td>Reads the date value in the specified locale, and then converts the date value to the local SAS date value.</td>
</tr>
<tr>
<td></td>
<td>NLDATEWw. Informat (p. 549)</td>
<td>Reads the date value in the specified locale and then converts the date value to the local SAS date and the day of the week.</td>
</tr>
<tr>
<td></td>
<td>NLDATMw. Informat (p. 550)</td>
<td>Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value.</td>
</tr>
<tr>
<td></td>
<td>NLDATMAP Informat (p. 551)</td>
<td>Reads the date value in the specified locale, and then converts the date value to the local SAS datetime with either a.m. or p.m.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLDATMWw. Informat (p. 552)</td>
<td>Reads the date value in the specified locale and then converts the date value to the local SAS day of the week and the datetime.</td>
<td></td>
</tr>
<tr>
<td>NLMNIAEDw.d Informat (p. 553)</td>
<td>Reads the monetary format of the international expression for the United Arab Emirates.</td>
<td></td>
</tr>
<tr>
<td>NLMNIAUDw.d Informat (p. 554)</td>
<td>Reads the monetary format of the international expression for Australia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIBGNw.d Informat (p. 555)</td>
<td>Reads the monetary format of the international expression for Bulgaria.</td>
<td></td>
</tr>
<tr>
<td>NLMNIBRLw.d Informat (p. 556)</td>
<td>Reads the monetary format of the international expression for Brazil.</td>
<td></td>
</tr>
<tr>
<td>NLMNICADw.d Informat (p. 557)</td>
<td>Reads the monetary format of the international expression for Canada.</td>
<td></td>
</tr>
<tr>
<td>NLMNICHFw.d Informat (p. 558)</td>
<td>Reads the monetary format of the international expression for Liechtenstein and Switzerland.</td>
<td></td>
</tr>
<tr>
<td>NLMNICNYw.d Informat (p. 559)</td>
<td>Reads the monetary format of the international expression for China.</td>
<td></td>
</tr>
<tr>
<td>NLMNICZKw.d Informat (p. 560)</td>
<td>Reads the monetary format of the international expression for the Czech Republic.</td>
<td></td>
</tr>
<tr>
<td>NLMNIDKKW.d Informat (p. 561)</td>
<td>Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
<td></td>
</tr>
<tr>
<td>NLMNIEEKw.d Informat (p. 562)</td>
<td>Reads the monetary format of the international expression for Estonia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIEGPw.d Informat (p. 563)</td>
<td>Reads the monetary format of the international expression for Egypt.</td>
<td></td>
</tr>
<tr>
<td>NLMNIEURw.d Informat (p. 564)</td>
<td>Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
<td></td>
</tr>
<tr>
<td>NLMNIGBPw.d Informat (p. 565)</td>
<td>Reads the monetary format of the international expression for the United Kingdom.</td>
<td></td>
</tr>
<tr>
<td>NLMNIHKDw.d Informat (p. 566)</td>
<td>Reads the monetary format of the international expression for Hong Kong.</td>
<td></td>
</tr>
<tr>
<td>NLMNIHKRW.d Informat (p. 567)</td>
<td>Reads the monetary format of the international expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIHUFw.d Informat (p. 568)</td>
<td>Reads the monetary format of the international expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLMNIIDRw.d Informat (p. 569)</td>
<td>Reads the monetary format of the international expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIILSw.d Informat (p. 570)</td>
<td>Reads the monetary format of the international expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNIINRw.d Informat (p. 571)</td>
<td>Reads the monetary format of the international expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNIJPYw.d Informat (p. 572)</td>
<td>Reads the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>NLMNIKRWw.d Informat (p. 573)</td>
<td>Reads the monetary format of the international expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNILTlw.d Informat (p. 574)</td>
<td>Reads the monetary format of the international expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNILVLw.d Informat (p. 575)</td>
<td>Reads the monetary format of the international expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIMOPw.d Informat (p. 576)</td>
<td>Reads the monetary format of the international expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>NLMNIEXNW.d Informat (p. 577)</td>
<td>Reads the monetary format of the international expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>NLMNIMYRW.d Informat (p. 578)</td>
<td>Reads the monetary format of the international expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>NLMNINOw.d Informat (p. 579)</td>
<td>Reads the monetary format of the international expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMNIIZWDw.d Informat (p. 580)</td>
<td>Reads the monetary format of the international expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNIPLNW.d Informat (p. 581)</td>
<td>Reads the monetary format of the international expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNIRUBW.d Informat (p. 582)</td>
<td>Reads the monetary format of the international expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>NLMNISEKw.d Informat (p. 583)</td>
<td>Reads the monetary format of the international expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>NLMNISGDw.d Informat (p. 584)</td>
<td>Reads the monetary format of the international expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>NLMNITHBW.d Informat (p. 585)</td>
<td>Reads the monetary format of the international expression for Thailand.</td>
</tr>
<tr>
<td></td>
<td>NLMNITRYW.d Informat (p. 586)</td>
<td>Reads the monetary format of the international expression for Turkey.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLMNITWDw.d Informat (p. 587)</td>
<td>Reads the monetary format of the international expression for Taiwan.</td>
</tr>
<tr>
<td></td>
<td>NLMNIUSDw.d Informat (p. 588)</td>
<td>Reads the monetary format of the international expression for Puerto Rico and the United States.</td>
</tr>
<tr>
<td></td>
<td>NLMNIZARw.d Informat (p. 589)</td>
<td>Reads the monetary format of the international expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>NLMNLAEDw.d Informat (p. 590)</td>
<td>Reads the monetary format of the local expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>NLMNLAUDw.d Informat (p. 591)</td>
<td>Reads the monetary format of the local expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBGNw.d Informat (p. 592)</td>
<td>Reads the monetary format of the local expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBRLw.d Informat (p. 593)</td>
<td>Reads the monetary format of the local expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCADw.d Informat (p. 594)</td>
<td>Reads the monetary format of the local expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCNFw.d Informat (p. 595)</td>
<td>Reads the monetary format of the local expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCNYw.d Informat (p. 596)</td>
<td>Reads the monetary format of the local expression for China.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCHKw.d Informat (p. 597)</td>
<td>Reads the monetary format of the local expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>NLMNLDDLw.d Informat (p. 598)</td>
<td>Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEEKw.d Informat (p. 599)</td>
<td>Reads the monetary format of the local expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEGKw.d Informat (p. 600)</td>
<td>Reads the monetary format of the local expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEURw.d Informat (p. 601)</td>
<td>Reads the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEGKw.d Informat (p. 602)</td>
<td>Reads the monetary format of the local expression for Hong Kong.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLMNLHRKw.d Informat (p. 604)</td>
<td>Reads the monetary format of the local expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHUFw.d Informat (p. 605)</td>
<td>Reads the monetary format of the local expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>NLMNLDIRw.d Informat (p. 606)</td>
<td>Reads the monetary format of the local expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLILSw.d Informat (p. 607)</td>
<td>Reads the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNLINRw.d Informat (p. 608)</td>
<td>Reads the monetary format of the local expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNLJPYw.d Informat (p. 609)</td>
<td>Reads the monetary format of the local expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLRWWw.d Informat (p. 610)</td>
<td>Reads the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLTLw.d Informat (p. 611)</td>
<td>Reads the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLVLw.d Informat (p. 612)</td>
<td>Reads the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMOPw.d Informat (p. 613)</td>
<td>Reads the monetary format of the local expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMXNw.d Informat (p. 614)</td>
<td>Reads the monetary format of the local expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMYRw.d Informat (p. 615)</td>
<td>Reads the monetary format of the local expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNOKw.d Informat (p. 616)</td>
<td>Reads the monetary format of the local expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNZDw.d Informat (p. 617)</td>
<td>Reads the monetary format of the local expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNLPNLw.d Informat (p. 618)</td>
<td>Reads the monetary format of the local expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLRUBw.d Informat (p. 619)</td>
<td>Reads the monetary format of the local expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLESEKw.d Informat (p. 620)</td>
<td>Reads the monetary format of the local expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>NLMNLSGDw.d Informat (p. 621)</td>
<td>Reads the monetary format of the local expression for Singapore.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLMNLTHBw.d Informat (p. 622)</td>
<td>Reads the monetary format of the local expression for Thailand.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTRYw.d Informat (p. 623)</td>
<td>Reads the monetary format of the local expression for Turkey.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTWDw.d Informat (p. 624)</td>
<td>Reads the monetary format of the local expression for Taiwan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLUSDw.d Informat (p. 625)</td>
<td>Reads the monetary format of the local expression for Puerto Rico, and the United States.</td>
</tr>
<tr>
<td></td>
<td>NLMNLZARw.d Informat (p. 626)</td>
<td>Reads the monetary format of the local expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>NLMNYw.d Informat (p. 628)</td>
<td>Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>NLMNYlw.d Informat (p. 629)</td>
<td>Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>NLNUMw.d Informat (p. 630)</td>
<td>Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>NLNUMlw.d Informat (p. 632)</td>
<td>Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>NLPCTw.d Informat (p. 633)</td>
<td>Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>NLPCTlw.d Informat (p. 635)</td>
<td>Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>NLSHYMON Informat (p. 636)</td>
<td>Reads the month name in the specified locale and converts it to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>NLTIMAPw. Informat (p. 637)</td>
<td>Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
<tr>
<td></td>
<td>NLTIMEw. Informat (p. 638)</td>
<td>Reads the time value in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
</tbody>
</table>

<p>| Character | $REVERJw. Informat (p. 639) | Reads character data from right to left and preserves blanks. |
|          | $REVERSw. Informat (p. 640) | Reads character data from right to left, and then left aligns the text. |
|          | $UCS2Bw. Informat (p. 641) | Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session. |</p>
<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UCS2BEw. Informat (p. 642)</td>
<td></td>
<td>Reads a character string that is in the encoding of the current SAS session and then converts the character string to big-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td>$UCS2Lw. Informat (p. 643)</td>
<td></td>
<td>Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UCS2LEw. Informat (p. 645)</td>
<td></td>
<td>Reads a character string that is in the encoding of the current SAS session and then converts the character string to little-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td>$UCS2Xw. Informat (p. 646)</td>
<td></td>
<td>Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UCS2XEw. Informat (p. 647)</td>
<td></td>
<td>Reads a character string that is in the encoding of the current SAS session and then converts the character string to 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td>$UCS4Bw. Informat (p. 648)</td>
<td></td>
<td>Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UCS4Lw. Informat (p. 649)</td>
<td></td>
<td>Reads a character string that is encoded in little-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UCS4Xw. Informat (p. 650)</td>
<td></td>
<td>Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UCS4XEw. Informat (p. 651)</td>
<td></td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to 32-bit, UCS4, Unicode encoding.</td>
</tr>
<tr>
<td>$UESCw. Informat (p. 652)</td>
<td></td>
<td>Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UESCEw. Informat (p. 654)</td>
<td></td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UESC representation.</td>
</tr>
<tr>
<td>$UNCRw. Informat (p. 655)</td>
<td></td>
<td>Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>$UNCREw. Informat (p. 656)</td>
<td></td>
<td>Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.</td>
</tr>
<tr>
<td>$UPARENw. Informat (p. 657)</td>
<td></td>
<td>Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Language Elements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$UPARENEw. Informat</td>
<td>(p. 658)</td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UPAREN representation.</td>
</tr>
<tr>
<td>$UPARENpw. Informat</td>
<td>(p. 660)</td>
<td>Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session, with national characters that remain in the encoding of the UPAREN representation.</td>
</tr>
<tr>
<td>$UTF8Xw. Informat</td>
<td>(p. 661)</td>
<td>Reads a character string that is encoded in UTF-8, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td><strong>Date and Time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JENGOw. Informat</td>
<td>(p. 540)</td>
<td>Reads Japanese kanji date values in the form yymmdd.</td>
</tr>
<tr>
<td>MINGUOw. Informat</td>
<td>(p. 546)</td>
<td>Reads dates in Taiwanese format.</td>
</tr>
<tr>
<td>NENGOw. Informat</td>
<td>(p. 547)</td>
<td>Reads Japanese date values in the form eyymmdd.</td>
</tr>
<tr>
<td>NLDATEw. Informat</td>
<td>(p. 548)</td>
<td>Reads the date value in the specified locale, and then converts the date value to the local SAS date value.</td>
</tr>
<tr>
<td>NLDATEWw. Informat</td>
<td>(p. 549)</td>
<td>Reads the date value in the specified locale and then converts the date value to the local SAS date and the day of the week.</td>
</tr>
<tr>
<td>NLDATMw. Informat</td>
<td>(p. 550)</td>
<td>Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value.</td>
</tr>
<tr>
<td>NLDATMAP Informat</td>
<td>(p. 551)</td>
<td>Reads the date value in the specified locale, and then converts the date value to the local SAS datetime with either a.m. or p.m.</td>
</tr>
<tr>
<td>NLDATEMW. Informat</td>
<td>(p. 552)</td>
<td>Reads the date value in the specified locale and then converts the date value to the local SAS day of the week and the datetime.</td>
</tr>
<tr>
<td>NLTIMAPw. Informat</td>
<td>(p. 637)</td>
<td>Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
<tr>
<td>NLTIMEw. Informat</td>
<td>(p. 638)</td>
<td>Reads the time value in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
<tr>
<td><strong>DBCS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$KANJ1w. Informat</td>
<td>(p. 541)</td>
<td>Removes shift code data from DBCS data.</td>
</tr>
<tr>
<td>$KANJXw. Informat</td>
<td>(p. 542)</td>
<td>Adds shift-code data to DBCS data.</td>
</tr>
<tr>
<td><strong>Hebrew Text Handling</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$CPTDWw. Informat</td>
<td>(p. 535)</td>
<td>Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>$\text{CPTWDw}$. Informat (p. 536)</td>
<td>Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding.</td>
</tr>
<tr>
<td>Numeric</td>
<td>$\text{EUROw.d Informat}$ (p. 537)</td>
<td>Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.</td>
</tr>
<tr>
<td></td>
<td>$\text{EUROXw.d Informat}$ (p. 539)</td>
<td>Reads numeric values and removes embedded characters in European currency.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIAEDw.d Informat}$ (p. 553)</td>
<td>Reads the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIAUDw.d Informat}$ (p. 554)</td>
<td>Reads the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIBGNw.d Informat}$ (p. 555)</td>
<td>Reads the monetary format of the international expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIBRLw.d Informat}$ (p. 556)</td>
<td>Reads the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNICADw.d Informat}$ (p. 557)</td>
<td>Reads the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNICFWw.d Informat}$ (p. 558)</td>
<td>Reads the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNICNYw.d Informat}$ (p. 559)</td>
<td>Reads the monetary format of the international expression for China.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNICZW.w.d Informat}$ (p. 560)</td>
<td>Reads the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIDKKw.d Informat}$ (p. 561)</td>
<td>Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIEEKw.d Informat}$ (p. 562)</td>
<td>Reads the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIEEGPw.d Informat}$ (p. 563)</td>
<td>Reads the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIEEURw.d Informat}$ (p. 564)</td>
<td>Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIGBPw.d Informat}$ (p. 565)</td>
<td>Reads the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td>$\text{NLMNIIKDW.d Informat}$ (p. 566)</td>
<td>Reads the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLMNIHRKw.d Informat (p. 567)</td>
<td>Reads the monetary format of the international expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIHUFw.d Informat (p. 568)</td>
<td>Reads the monetary format of the international expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>NLMNIIDRw.d Informat (p. 569)</td>
<td>Reads the monetary format of the international expression for Indonesia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIILSw.d Informat (p. 570)</td>
<td>Reads the monetary format of the international expression for Israel.</td>
<td></td>
</tr>
<tr>
<td>NLMNIINRw.d Informat (p. 571)</td>
<td>Reads the monetary format of the international expression for India.</td>
<td></td>
</tr>
<tr>
<td>NLMNIPYw.d Informat (p. 572)</td>
<td>Reads the monetary format of the international expression for Japan.</td>
<td></td>
</tr>
<tr>
<td>NLMNKRw.d Informat (p. 573)</td>
<td>Reads the monetary format of the international expression for South Korea.</td>
<td></td>
</tr>
<tr>
<td>NLMNILTLw.d Informat (p. 574)</td>
<td>Reads the monetary format of the international expression for Lithuania.</td>
<td></td>
</tr>
<tr>
<td>NLMNILVLw.d Informat (p. 575)</td>
<td>Reads the monetary format of the international expression for Latvia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMOPw.d Informat (p. 576)</td>
<td>Reads the monetary format of the international expression for Macau.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMXNw.d Informat (p. 577)</td>
<td>Reads the monetary format of the international expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMYRw.d Informat (p. 578)</td>
<td>Reads the monetary format of the international expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>NLMNINOKw.d Informat (p. 579)</td>
<td>Reads the monetary format of the international expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>NLMNINZDw.d Informat (p. 580)</td>
<td>Reads the monetary format of the international expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>NLMNIPLNw.d Informat (p. 581)</td>
<td>Reads the monetary format of the international expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>NLMNIRUBw.d Informat (p. 582)</td>
<td>Reads the monetary format of the international expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>NLMNISEKw.d Informat (p. 583)</td>
<td>Reads the monetary format of the international expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>NLMNISGDw.d Informat (p. 584)</td>
<td>Reads the monetary format of the international expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLMNITHBw.d Informat (p. 585)</td>
<td></td>
<td>Reads the monetary format of the international expression for Thailand.</td>
</tr>
<tr>
<td>NLMNITRYw.d Informat (p. 586)</td>
<td></td>
<td>Reads the monetary format of the international expression for Turkey.</td>
</tr>
<tr>
<td>NLMNITWDw.d Informat (p. 587)</td>
<td></td>
<td>Reads the monetary format of the international expression for Taiwan.</td>
</tr>
<tr>
<td>NLMNIUSDw.d Informat (p. 588)</td>
<td></td>
<td>Reads the monetary format of the international expression for Puerto Rico and the United States.</td>
</tr>
<tr>
<td>NLMNIZARw.d Informat (p. 589)</td>
<td></td>
<td>Reads the monetary format of the international expression for South Africa.</td>
</tr>
<tr>
<td>NLMNLAEDw.d Informat (p. 590)</td>
<td></td>
<td>Reads the monetary format of the local expression for the United Arab Emirates.</td>
</tr>
<tr>
<td>NLMNLAUDw.d Informat (p. 591)</td>
<td></td>
<td>Reads the monetary format of the local expression for Australia.</td>
</tr>
<tr>
<td>NLMNLBGNw.d Informat (p. 592)</td>
<td></td>
<td>Reads the monetary format of the local expression for Bulgaria.</td>
</tr>
<tr>
<td>NLMNLBRLw.d Informat (p. 593)</td>
<td></td>
<td>Reads the monetary format of the local expression for Brazil.</td>
</tr>
<tr>
<td>NLMNLCADw.d Informat (p. 594)</td>
<td></td>
<td>Reads the monetary format of the local expression for Canada.</td>
</tr>
<tr>
<td>NLMNLCFWw.d Informat (p. 595)</td>
<td></td>
<td>Reads the monetary format of the local expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td>NLMNLCNYw.d Informat (p. 596)</td>
<td></td>
<td>Reads the monetary format of the local expression for China.</td>
</tr>
<tr>
<td>NLMNLCZKw.d Informat (p. 597)</td>
<td></td>
<td>Reads the monetary format of the local expression for the Czech Republic.</td>
</tr>
<tr>
<td>NLMNLDKKw.d Informat (p. 598)</td>
<td></td>
<td>Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.</td>
</tr>
<tr>
<td>NLMNLLEEKw.d Informat (p. 599)</td>
<td></td>
<td>Reads the monetary format of the local expression for Estonia.</td>
</tr>
<tr>
<td>NLMNLEGpw.d Informat (p. 600)</td>
<td></td>
<td>Reads the monetary format of the local expression for Egypt.</td>
</tr>
<tr>
<td>NLMNLEURw.d Informat (p. 601)</td>
<td></td>
<td>Reads the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLMNLGBPw.d Informat (p. 602)</td>
<td>Reads the monetary format of the local expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHKDw.d Informat (p. 603)</td>
<td>Reads the monetary format of the local expression for Hong Kong.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHRKw.d Informat (p. 604)</td>
<td>Reads the monetary format of the local expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHUFw.d Informat (p. 605)</td>
<td>Reads the monetary format of the local expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>NLMNLIDRw.d Informat (p. 606)</td>
<td>Reads the monetary format of the local expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLILSw.d Informat (p. 607)</td>
<td>Reads the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLINRw.d Informat (p. 608)</td>
<td>Reads the monetary format of the local expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLJPYw.d Informat (p. 609)</td>
<td>Reads the monetary format of the local expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLKRWw.d Informat (p. 610)</td>
<td>Reads the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLTLw.d Informat (p. 611)</td>
<td>Reads the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLVLw.d Informat (p. 612)</td>
<td>Reads the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMOPw.d Informat (p. 613)</td>
<td>Reads the monetary format of the local expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMXNw.d Informat (p. 614)</td>
<td>Reads the monetary format of the local expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMYRw.d Informat (p. 615)</td>
<td>Reads the monetary format of the local expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>NLMNMLNOKw.d Informat (p. 616)</td>
<td>Reads the monetary format of the local expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNZDw.d Informat (p. 617)</td>
<td>Reads the monetary format of the local expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNLPNw.d Informat (p. 618)</td>
<td>Reads the monetary format of the local expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLRUBw.d Informat (p. 619)</td>
<td>Reads the monetary format of the local expression for Russia.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLMNLSEKw.d Informat (p. 620)</td>
<td>Reads the monetary format of the local expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>NLMNLSGDw.d Informat (p. 621)</td>
<td>Reads the monetary format of the local expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTBHw.d Informat (p. 622)</td>
<td>Reads the monetary format of the local expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTRYw.d Informat (p. 623)</td>
<td>Reads the monetary format of the local expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTWDw.d Informat (p. 624)</td>
<td>Reads the monetary format of the local expression for Taiwan.</td>
<td></td>
</tr>
<tr>
<td>NLMNLUUSDw.d Informat (p. 625)</td>
<td>Reads the monetary format of the local expression for Puerto Rico, and the United States.</td>
<td></td>
</tr>
<tr>
<td>NLMNLZARw.d Informat (p. 626)</td>
<td>Reads the monetary format of the local expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>NLMNYw.d Informat (p. 628)</td>
<td>Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLMNYIw.d Informat (p. 629)</td>
<td>Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLNUMw.d Informat (p. 630)</td>
<td>Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLNUMIw.d Informat (p. 632)</td>
<td>Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLPCTw.d Informat (p. 633)</td>
<td>Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLPCTIw.d Informat (p. 635)</td>
<td>Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLSTRMON Informat (p. 636)</td>
<td>Reads the month name in the specified locale and converts it to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>YENw.d Informat (p. 665)</td>
<td>Removes embedded yen signs, commas, and decimal points.</td>
<td></td>
</tr>
</tbody>
</table>
Dictionary

$CPTD_{Ww}$. Informat

Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Hebrew Text Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>This informat is not supported in a DATA step that runs in CAS.</td>
</tr>
</tbody>
</table>

Syntax

$CPTD_{Ww}$.

Syntax Description

$w$

specifies the width of the input field.

Default 200

Range 1–32767

Comparisons

The $CPTD_{Ww}$. informat performs processing that is opposite of the $CPTWD_{Ww}$. informat.

Example

The following example uses the input value of 808182.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('808182', $cptd6.); put x;</td>
<td>י&quot;ע</td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$CPTD_{Ww}$. Format" on page 123
$CPTWDw. Informat

Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding.

Category: Hebrew Text Handling

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$CPTWDw.

Syntax Description

w

specifies the width of the input field.

Default 200

Range 1–32767

Comparisons

The $CPTWDw. informat performs processing that is opposite of the $CPTDWw. informat.

Example

The following example uses the input value of 存档.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('存档', $cptwd6.);</td>
<td>$□,</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
EUROw.d Informat

Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.

Category: Numeric
Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

EUROw.d

Syntax Description

\(w\)

specifies the width of the input field.

Default 6

Range 1–32

\(d\)

specifies the power of 10 by which to divide the value. If the data contains decimal points, the \(d\) value is ignored.

Default 0

Range 0–31

Details

The EUROw.d informat reads numeric values and removes embedded euro symbols (E), commas, blanks, percent signs, hyphens, and close parentheses from the input data. A decimal point is assumed to be a separator between the whole number and the decimal portion. The EUROw.d informat converts an open parenthesis at the beginning of a field to a minus sign.

Comparisons

The EUROw.d informat is similar to the EUROXw.d informat, but EUROXw.d reverses the roles of the decimal point and the comma. This convention is common in European countries.
If no commas or periods appear in the input, then the EUROw.d and the EUROXw.d informats are interchangeable.

Example

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

```sas
data _null_
  input x euro10.;
  put x=
  datalines;
E1
E1.23
1.23
1,234.56
;
run;
SAS Log:
x=1
x=1.23
x=1.23
x=1234.56
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>input x euro10.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>E1.23</td>
<td>input x euro10.;</td>
<td>1.23</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1.23</td>
<td>input x euro10.;</td>
<td>1.23</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1,234.56</td>
<td>input x euro10.;</td>
<td>1,234.56</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

**Formats:**

- “EUROw.d Format” on page 132
- “EUROXw.d Format” on page 135

**Informat:**

- “EUROXw.d Informat” on page 539
EUROXw.d Informat

Reads numeric values and removes embedded characters in European currency.

Category: Numeric

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

EUROXw.d

Syntax Description

w

specifies the width of the input field.

Default 6

Range 1–32

d

specifies the power of 10 by which to divide the value. If the data contains a comma, which represents a decimal point, the d value is ignored.

Default 0

Range 0–31

Details

The EUROXw.d informat reads numeric values and removes embedded euro symbols (E), periods, blanks, percent signs, hyphens, and close parentheses from the input data. A comma is assumed to be a separator between the whole number and the decimal portion. The EUROXw.d informat converts an open parenthesis at the beginning of a field to a minus sign.

Comparisons

- The EUROXw.d informat is similar to the EUROw.d informat, but EUROw.d reverses the roles of the comma and the decimal point. This convention is common in English-speaking countries.

- If no commas or periods appear in the input, the EUROXw.d and the EUROw.d informats are interchangeable.

Example

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

data _null_;
input x eurox10.;
put x=
datalines;

B1
B1.23
1.23
1,234.56
; run;
SAS Log:
7    input x eurox10.;
8    put x=
9    datalines;
x=1
x=123
x=123
x=1.23456

<table>
<thead>
<tr>
<th>Values</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| B1     | input x eurox10.;
        | put x;      | 1       |
| B1.23  | input x eurox10.;
        | put x;      | 123     |
| 1.23   | input x eurox10.;
        | put x;      | 123     |
| 1,234.56 | input x eurox10.;
           | put x;      | 1.23456 |

See Also

Formats:
- "EUROw.d Format" on page 132
- "EUROXw.d Format" on page 135

Informat:
- "EUROw.d Informat" on page 537

JNENGOw. Informat
Reads Japanese kanji date values in the form yyymmdd.
Category: Date and Time
This informat is not supported in a DATA step that runs in CAS.

Syntax

\texttt{JNENGO}w.

Syntax Description

\texttt{w}

specifies the width of the input field.

Default 16

Range 16–32

Details

The \texttt{JNENGO}w. informat reads Japanese kanji values in the form \texttt{yymmdd}. You can separate the year, month, and day values by blanks or special characters. The width of the input field should include space for blanks and special characters.

Note: SAS interprets a two-digit year as belonging to the 100-year span that is defined by the \texttt{YEARCUTOFF=} system option.

Example

The following example shows how to use the \texttt{JNENGO} informat to convert kanji values to SAS date values. The date values in the data lines are separated by special characters.

\begin{verbatim}
data _null_;  
   input x JNENGO.;  
data =lines;
明治1年4月6日  
明治45年7月29日  
大正1年7月30日  
大正15年12月24日  
昭和1年12月25日  
昭和64年1月7日  
平成1年1月8日  
平成10年12月8日;
\end{verbatim}

$\texttt{KANJ}\texttt{J}w. Informat$

Removes shift code data from DBCS data.
$KANJIw.

Syntax Description

\( w \)

- specifies the width of the input field.

Range

The minimum width for the informat is 2.

Restriction

The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.

Details

The $KANJI informat removes shift-code data from DBCS data. The $KANJI informat processes host-mainframe data. $KANJI can be used on other platforms. If you use the $KANJI informat on non-EBCDIC (non-modal encoding) hosts, the data does not change.

The data must start with SO and end with SI, unless single-byte blank data are returned. The input data length must be \( 2 + (SO/SI \text{ length})*2 \).

See Also

Formats:

- "$KANJIw. Format" on page 144
- "$KANJIIXw. Format" on page 145

Informat:

- "$KANJIIXw. Informat" on page 542

$KANJIIXw. Informat

Adds shift-code data to DBCS data.

Syntax

$KANJIIXw.
Syntax Description

```
w
```
specifies the width of the input field.

**Range**
The minimum width for the informat is \(2 + (\text{length of shift code used on the current DBCSTYPE= setting}) \times 2\).

**Restriction**
The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.

Details

The $KANJIX$ informat adds shift-code data to DBCS data that does not have shift-code data. If the input data is blank, shift-code data is not added. The $KANJIX$ informat processes host-mainframe data, but $KANJIX$ can be used on other platforms. If you use the $KANJIX$ informat on non-EBCDIC (non-modal encoding) hosts, the data does not change.

See Also

**Formats:**
- “$KANJIw. Format” on page 144
- “$KANJIXw. Format” on page 145

**Informat:**
- “$KANJIw. Informat” on page 541

$LOGVSw. Informat

Reads a character string that is in left-to-right logical order, and then converts the character string to visual order.

**Category:** BIDI Text Handling

**Restriction:** This informat is not supported in a DATA step that runs in CAS.

Syntax

```
$LOGVSw.
```

Syntax Description

```
w
```
specifies the width of the input field.

**Default** 200
Range 1–32767

Comparisons
The $LOGVS_{w}$. informat performs processing that is opposite to the LOGVSR_{w}.
informat.

Example
The following example uses the Hebrew input value of "نظير" flight."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('نظير flight',logvs12.); put x;</td>
<td>נתיי תונק flight</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "پرست" computer."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('پرست computer',logvs12.); put x;</td>
<td>دنات پرست computer</td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$LOGVSRw. Format" on page 148
- "$LOGVSw. Format" on page 146

Informat:

- "$LOGVSRw. Informat" on page 544

$LOGVSRw. Informat
Reads a character string that is in right-to-left logical order, and then converts the character string to visual order.
Syntax

$LOGVSRw.

Syntax Description

w

specifies the width of the input field.

Default 200

Range 1–32767

Comparisons

The $LOGVSRw. informat performs processing that is opposite to the $LOGVSw. informat.

Example

The following example uses the Hebrew input value of "flight."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('ף seeker',logv12.); put x;</td>
<td>flight</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "computer."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('אדר݂ computer',logv12.); put x;</td>
<td>computer</td>
</tr>
</tbody>
</table>

See Also

Formats:
MINGUOW. Informat

Reads dates in Taiwanese format.

Category: Date and Time

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

MINGUOW.

Syntax Description

w
specifies the width of the input field.

Default 6

Range 6–10

Details

The general form of a Taiwanese date is yyyyymmdd:

yyyy
is an integer that represents the year.

mm
is an integer from 01 through 12 that represents the month.

dd
is an integer from 01 through 31 that represents the day of the month.

The Taiwanese calendar uses 1912 as the base year (01/01/01 is January 1, 1912). Dates before 1912 are not valid. Year values do not roll over after 100 years. Instead, they continue to increase.

You can separate the year, month, and day values with any delimiters, such as blanks, slashes, or hyphens, that are permitted by the YYMMDDw. informat. If delimiters are used, place them between all the values. If you omit delimiters, be sure to use a leading zero for days or months that have a value less than 10.

Example

The following examples use different dates for input values.
input date minguo10.;
put date date9.;
data _null_;  
  input date minguo10.;
  put date date9.;
datalines;
49/01/01
891215
03-01-01;

Values               Results

49/01/01             01JAN1960
891215               15DEC2000
103-01-01            01JAN2014

See Also

Format:

- "MINGUOw. Format" on page 149

Informat:

- "YYMMDDw. Informat" in SAS Formats and Informats: Reference

NENGOw. Informat

Reads Japanese date values in the form eyymmdd.

Category:          Date and Time
Restriction:       This informat is not supported in a DATA step that runs in CAS.

Syntax

NENGOw.

Syntax Description

w

specifies the width of the input field.

Default   10
Details

The general form of a Japanese date is eyymmdd:

- **e** is the first letter of the name of the imperial era (Meiji, Taisho, Showa, Heisei, or Reiwa).
- **yy** is an integer that represents the year.
- **mm** is an integer from 01 through 12 that represents the month.
- **dd** is an integer from 01 through 31 that represents the day of the month.

The e value can be separated from the integers by a period. If you omit e, SAS uses the current imperial era. You can separate the year, month, and day values by blanks or any nonnumeric character. However, if delimiters are used, place them between all the values. If you omit delimiters, be sure to use a leading zero for days or months that are values less than 10.

Example

The following example uses different input values.

```sas
data _null_;  
input x nengo.;  
put x= date9.;  
datalines;  
h11108  
h.11108  
11/10/08  
;  
run;
```

x=08OCT1999  
x=08OCT1999  
x=08OCT1999  
x=08OCT1999

See Also

**Formats:**

* "NENGOw. Format" on page 151

**NLDATEw. Informat**

Reads the date value in the specified locale, and then converts the date value to the local SAS date value.

Categories: CAS
Date and Time

Alias:
NLDATEW

Syntax
NLDATEw.

Syntax Description
w
specifies the width of the input field.

Default 10
Range 10–200

Example
The following examples use the input February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>15760</td>
</tr>
<tr>
<td>dy='February 24, 2003';</td>
<td></td>
</tr>
<tr>
<td>y=input('dy,nldate200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>15760</td>
</tr>
<tr>
<td>dy='24. Februar 2003';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldate16.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLDATEw. Format" on page 154

NLDATEWw. Informat
Reads the date value in the specified locale and then converts the date value to the local SAS date and the day of the week.

Categories: CAS
Syntax

**NLDATETIMEw.**

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 20

Range 16–200

Example

The following examples use the input February 24, 2014.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>19778</td>
</tr>
<tr>
<td>data;</td>
<td></td>
</tr>
<tr>
<td>dy='Monday, February 24, 2014';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldatew200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>19778</td>
</tr>
<tr>
<td>dy='Mo. 24.Februar 2014';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldatew16.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLDATETIMEw. Informat**

Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value.

Categories: CAS  
Date and Time

Alias: NLDATMW, NLDATMAP
Syntax

NLDATMw.

Syntax Description

w

specifies the width of the input field.

Default 19

Range 19–200

Example

The following examples use the input value of February 24, 2003 12:39:43.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>1361709583</td>
</tr>
<tr>
<td>y=input('24.Feb03:12:39:43', nldatm.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>1330171200</td>
</tr>
<tr>
<td>y=input('24.Februar 2003 12.39 Uhr', nldatm.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLDATMw. Format" on page 176

NLDATMAP Informat

Reads the date value in the specified locale, and then converts the date value to the local SAS datetime with either a.m. or p.m.

Categories: CAS
Date and Time

Alignment: Left
Syntax

\textbf{NLDATMAP}_w.

Syntax Description

\textit{w} specifies the width of the output field. If necessary, SAS abbreviates the datetime value to fit the format width.

\textbf{Default} 32

\textbf{Range} 17–200

Example

These examples use the input value of 12:39:43 p.m. on February 24, 2014.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| \begin{verbatim}
options locale=English_UnitedStates;
data;
dy='February 24, 2014 12:39:43 PM';
y=input(dy,nldatmap200.);
put y=;
run;
\end{verbatim} | 1708864783 |

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| \begin{verbatim}
options locale=Spanish_Mexico;
data;
dy='24/02/2003 12:39:43 PM';
y=input(dy,nldatmap200.);
put y=;
run;
\end{verbatim} | 1708864783 |

\textbf{NLDATMW}_w. Informat

Reads the date value in the specified locale and then converts the date value to the local SAS day of the week and the datetime.

\textbf{Categories:} CAS
Date and Time

\textbf{Alignment:} Left

Syntax

\textbf{NLDATMW}_w
Syntax Description

**w**

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default: 40  
Range: 34–200

Example

The following examples use the input Mon, Feb 24, 2014 12:39:43 PM.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>data;</td>
<td></td>
</tr>
<tr>
<td>dy='Mon, Feb 24, 2014 12:39:43 PM';</td>
<td>1708864783</td>
</tr>
<tr>
<td>y=input(dy,nldatmw200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>dy='Mo, 24. Feb 2014 12.39 Uhr';</td>
<td>1708864783</td>
</tr>
<tr>
<td>y=input(dy,nldatmw16.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLMNIAEDw.d Informat**

Reads the monetary format of the international expression for the United Arab Emirates.

Categories: CAS  
Numeric  
Alignment: Left

Syntax

**NLMNIAEDw.d**

Syntax Description

**w**

specifies the width of the output field.

Default: 12
Range  8–32

d  specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.

Default  3  
Range  0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmiaed32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:
- "NLMNLAEDw.d Informat" on page 590

NLMNIAUDw.d Informat

Reads the monetary format of the international expression for Australia.

Categories:  CAS  
Numeric

Alignment:  Left

Syntax

NLMNIAUDw.d

Syntax Description

w  specifies the width of the output field.
Default 12
Range 8–32

\( d \)
specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 2
Range 0–28

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{input}('($12,345.67)', \text{nlniaud32.2});
y = \text{input}('($12,345.67)', \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

“NLMNLAUDw.d Informat” on page 591

NLMNIBGNw.d Informat
Reads the monetary format of the international expression for Bulgaria.

Categories: CAS
Numeric
Alignment: Left

Syntax

\[ \text{NLMNIBGN}w.d \]
Syntax Description

**w**

specifies the width of the output field.

Default 12
Range 8–32

**d**

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put('(-1234.56789)',nlmnibgn32.2);
y=put('(-1234.56789)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- "NLMNGBKGNw.d Informat" on page 592

---

### NLMNIBRLW.d Informat

Reads the monetary format of the international expression for Brazil.

**Categories:** CAS Numeric

**Alignment:** Left

**Syntax**

NLMNIBRLW.d
Syntax Description

**w**

specifies the width of the output field.

Default: 12  
Range: 8–32  

**d**

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 2  
Range: 0–28  

Example

In the following example, the LOCALE= system option is set to English UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnibrl32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- "NLMNLBRLw.d Informat" on page 593

---

**NLMNICADw.d Informat**

Reads the monetary format of the international expression for Canada.

Categories:  
- CAS  
- Numeric  

Alignment:  
- Left  

Syntax

**NLMNICADw.d**
### Syntax Description

**w**
- Specifies the width of the output field.
- **Default**: 12
- **Range**: 8–32

**d**
- Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
- **Default**: 2
- **Range**: 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnacid32.2);
y=input'($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

- **Format:**
  - "NLMNICADw.d Format" on page 207

---

### NLMNICHFw.d Informat

Reads the monetary format of the international expression for Liechtenstein and Switzerland.

- **Categories**: CAS, Numeric
- **Alignment**: Left

### Syntax

```
NLMNICHFw.d
```
Syntax Description

\( w \)

specifies the width of the output field.

Default 12
Range 8–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{input}(\text{'($12,345.67)'}, \text{nlmnicf32.2}); \]
\[ y = \text{input}(\text{'($12,345.67)'dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNICHFw.d Format" on page 208

NLMNICNYw.d Informat

Reads the monetary format of the international expression for China.

Categories: CAS Numeric
Alignment: Left

Syntax

\[ \text{NLMNICNY}w.d \]
Syntax Description

**w**
specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

**d**
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

- **Default**: 2
- **Range**: 0–28

Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```plaintext
x=input'($12,345.67)',nlmnicny32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- “**NLMNICNYw.d Format**” on page 209

---

**NLMNICZKw.d Informat**

Reads the monetary format of the international expression for the Czech Republic.

**Categories:**

- CAS
- Numeric

**Alignment:**

- Left

Syntax

**NLMNICZKw.d**
Syntax Description

\( w \)

specifies the width of the output field.

- Default: 12
- Range: 8–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- Default: 4
- Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{input('($12,345.67)$',nlmniczk32.2)}; \\
y &= \text{input('($12,345.67)$',dollar32.2)};
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- "NLMNLCCZKw.d Informat" on page 597

---

**NLMNIDKKw.d Informat**

Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.

- Categories: CAS, Numeric
- Alignment: Left

Syntax

\[ \text{NLMNIDKKw.d} \]
Syntax Description

\textbf{w}

specifies the width of the output field.

\begin{itemize}
\item Default \hspace{1cm} 12
\item Range \hspace{1cm} 8–32
\end{itemize}

\textbf{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

\begin{itemize}
\item Default \hspace{1cm} 2
\item Range \hspace{1cm} 0–28
\end{itemize}

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmndkk32.2);
y=input'($12,345.67)',dollar32.2);
\end{verbatim}

\begin{tabular}{|c|c|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
put x=; & -12345.67 \\
put y=; & -12345.67 \\
\hline
\end{tabular}

See Also

\textbf{Format:}

\begin{itemize}
\item “\texttt{NLMNIDKKw.d Format}” on page 211
\end{itemize}

\textbf{NLMNIEEKw.d Informat}

Reads the monetary format of the international expression for Estonia.

\begin{itemize}
\item Categories: \hspace{1cm} CAS
\item \hspace{1cm} Numeric
\end{itemize}

\begin{itemize}
\item Alignment: \hspace{1cm} Left
\end{itemize}

Syntax

\texttt{NLMNIEEKw.d}
Syntax Description

**w**

specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

**d**

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default**: 4
- **Range**: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnieek32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- "NLMNLEEKw.d Informat" on page 599

---

**NLMNIEGPw.d Informat**

Reads the monetary format of the international expression for Egypt.

**Categories:** CAS

**Numeric**

**Alignment:** Left

**Syntax**

```
NLMNIEGPw.d
```
Syntax Description

\[ w \]

specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

\[ d \]

d specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default**: 3
- **Range**: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmniegp32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

- **Informat**: "NLMNLEGPw.d Informat" on page 600

---

**NLMNIEURw.d Informat**

Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

- **Categories**: CAS, Numeric
- **Alignment**: Left
Syntax

**NLMNIEURw.d**

Syntax Description

**w**

specifies the width of the output field.

Default: 12

Range: 8–32

**d**

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 2

Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)',nlmnieur32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLMNIEURw.d Format" on page 214

**NLMNIGBPw.d Informat**

Reads the monetary format of the international expression for the United Kingdom.

Categories: CAS, Numeric

Alignment: Left
Syntax

\texttt{NLMNIGBPw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\textit{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 2

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input'($12,345.67)',nlmnigbp32.2);
y=input'($12,345.67)'dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=; } & \texttt{-12345.67} \\
\texttt{put y=; } & \texttt{-12345.67} \\
\hline
\end{tabular}

See Also

Format:

- \texttt{“NLMNIGBPw.d Format” on page 215}

\textbf{NLMNIHKDw.d Informat}

Reads the monetary format of the international expression for Hong Kong.

Categories: CAS, Numeric

Alignment: Left
Syntax

\texttt{NLMNIHKDw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default \hspace{1em} 12

Range \hspace{1em} 8–32

\textit{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default \hspace{1em} 2

Range \hspace{1em} 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input'($12,345.67)',nlmnihkd32.2);
y=input'($12,345.67)'dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\begin{itemize}
\item put x=;
\item put y=;
\end{itemize} & \begin{itemize}
\item -12345.67
\item -12345.67
\end{itemize} \\
\hline
\end{tabular}

See Also

\textbf{Format:}

\begin{itemize}
\item "\texttt{NLMNIHKDw.d Format}" on page 216
\end{itemize}
Syntax

**NLMNIHRK**\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default 12

Range 8–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English _UnitedStates_.

\[x=\text{input} \left('\left($12,345.67\right)'\right)\text{,nlmnihrk32.2}];\]

\[y=\text{input} \left('\left($12,345.67\right)'\right)\text{,dollar32.2}];\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- "**NLMNLHRK**\(w.d\) Informat" on page 604

---

**NLMNIHUF**\(w.d\) Informat

Reads the monetary format of the international expression for Hungary.

**Categories:**

- CAS
- Numeric

**Alignment:**

- Left
Syntax

NLMNIHUFw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnihuf32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:
- “NLMLNHUFW.d Informat” on page 605

NLMNIIDRw.d Informat

Reads the monetary format of the international expression for Indonesia.

Categories: CAS
Numeric

Alignment: Left
Syntax

**NLMNIIDRw.d**

Syntax Description

`w` specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

`d` specifies to divide the number by $10^d$. If the data contains decimal points, the `d` value is ignored.

- **Default**: 2
- **Range**: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x = input('($12,345.67)', nlmniidr32.2);
y = input('($12,345.67)', dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informats**:

- “NLMNLIDRw.d Informat” on page 606

---

**NLMNIILSw.d Informat**

Reads the monetary format of the international expression for Israel.

- **Categories**: CAS, Numeric
- **Alignment**: Left
Syntax

**NLMNIILSw.d**

Syntax Description

\( w \)

specifies the width of the output field.

- Default: 12
- Range: 8–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

- Default: 4
- Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)',nlmniils32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- “NLMNIILSw.d Format” on page 220

---

**NLMNIINRw.d Informat**

Reads the monetary format of the international expression for India.

Categories: CAS, Numeric

Alignment: Left
Syntax

\textbf{NLMNINR}w.d

Syntax Description

\textit{w} specifies the width of the output field.

Default: 12

Range: 8–32

\textit{d} specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.

Default: 2

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmniinr32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

\textbf{Informat}:

\begin{itemize}
  \item “NLMNLINRw.d Informat” on page 608
\end{itemize}
Syntax

\texttt{NLMNIJPY}\texttt{w.d}

Syntax Description

\textbf{w}

specifies the width of the output field.

Default \hspace{0.5cm} 12

Range \hspace{0.5cm} 8–32

\textbf{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default \hspace{0.5cm} 0

Range \hspace{0.5cm} 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input'('$(12,345.67)'\,\text{"nlmijpy"}32.2);\n
\end{verbatim}

\begin{verbatim}
y=input'('$(12,345.67)'\,\text{"dollar"}32.2);\n
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- “\texttt{NLMNIJPY}w.d Format” on page 222

\texttt{NLMNIKRW}w.d Informat

Reads the monetary format of the international expression for South Korea.

Categories: \hspace{1cm} CAS

Numeric

Alignment: \hspace{1cm} Left
Syntax

NLMNIKRWw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input'($12,345.67)',nlni32.2);
y=input'($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNLKRWw.d Informat” on page 610

NLMNILTLw.d Informat

Reads the monetary format of the international expression for Lithuania.

Categories: CAS, Numeric

Alignment: Left
Syntax

**NLMNILTLw.d**

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 4
  - Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input('($12,345.67)',nlniltl32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- "NLMNLLTLw.d Informat" on page 611
Syntax

NLMNIVLw.d

Syntax Description

w

specifies the width of the output field.

Default 12

Range 8–32

d

dspecifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 4

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnilvl32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNLLVLw.d Informat” on page 612

NLMNIMOPw.d Informat

Reads the monetary format of the international expression for Macau.

Categories: CAS

Numeric

Alignment: Left
Syntax

NLMNIMOPw.d

Syntax Description

w
specifies the width of the output field.

Default  12
Range     8–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default  2
Range     0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnimop32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNLMOPOw.d Informat” on page 613

NLMNIMXNw.d Informat

Reads the monetary format of the international expression for Mexico.

Categories: CAS
            Numeric

Alignment: Left
Syntax

NLMNIMXNw.d

Syntax Description

w  
specifies the width of the output field.
  Default  12
  Range   8–32

d  
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
  Default  2
  Range   0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnimxn32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- "NLMNIMXNw.d Informat" on page 614

NLMNIMYRw.d Informat

Reads the monetary format of the international expression for Malaysia.

Categories: CAS

Numeric

Alignment: Left
Syntax

\[ \text{NLMNIMYR}w.d \]

Syntax Description

\( w \)

specifies the width of the output field.

- Default: 12
- Range: 8–32

\( d \)

specifies to divide the number by 10\(^d\). If the data contains decimal points, the \( d \) value is ignored.

- Default: 2
- Range: 0–28

Example

In the following example, the LOCATE= system option is set to English_UnitedStates.

\[
\begin{align*}
x & = \text{input('($12,345.67)$',nlmnimyr32.2);} \\
y & = \text{input('($12,345.67)$',dollar32.2)};
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNIMYRw.d Format" on page 228

---

NLMNINOKw.d Informat

Reads the monetary format of the international expression for Norway.

Categories: CAS, Numeric

Alignment: Left
Syntax

NLMNINOK\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 12

Range 8–32

\textit{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=input'($12,345.67)',nlmninok32.2);
y=input'($12,345.67)'dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
put x=; & -12345.67 \\
put y=; & -12345.67 \\
\hline
\end{tabular}

See Also

Format:

- "NLMNINOKw.d Format" on page 229

NLMNINZDw.d Informat

Reads the monetary format of the international expression for New Zealand.

Categories: CAS

Numeric

Alignment: Left
Syntax

NLMNINZDw.d

Syntax Description

w
 specifies the width of the output field.
 Default 12
 Range 8–32

d
 specifies to divide the number by $10^d$. If the data contains decimal points, the $d$
 value is ignored.
 Default 2
 Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmninzd32.2);
y=input'($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNINZDw.d Format” on page 230

NLMNIPLNw.d Informat

Reads the monetary format of the international expression for Poland.

Categories: CAS
 Numeric

Alignment: Left
Syntax

\texttt{NLMNIPLN} \texttt{w.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default \hspace{1em} 12

Range \hspace{1em} 8–32

\texttt{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default \hspace{1em} 2

Range \hspace{1em} 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
  x=input'($12,345.67)',nlmnipln32.2);
  y=input'($12,345.67)'dollar32.2);
\end{verbatim}

\begin{center}
\begin{tabular}{ll}
\textbf{Statements} & \textbf{Results} \\
\hline
  put x=; & -12345.67 \\
  put y=; & -12345.67 \\
\end{tabular}
\end{center}

See Also

\textbf{Format:}

\begin{itemize}
  \item \texttt{"NLMNIPLNw.d Format" on page 231}
\end{itemize}

\textbf{NLMNIRUBw.d Informat}

Reads the monetary format of the international expression for Russia.

Categories: \hspace{1em} CAS \hspace{1em} Numeric

Alignment: \hspace{1em} Left
**Syntax**

`NLMNIRUB w.d`

**Syntax Description**

- **`w`** specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **`d`** specifies to divide the number by $10^d$. If the data contains decimal points, the `d` value is ignored.
  - Default: 2
  - Range: 0–28

**Example**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input'($12,345.67)',nlnirub32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

- **Format:**
  - “NLMNIRUBw.d Format” on page 232

**NLMNISEKw.d Informat**

Reads the monetary format of the international expression for Sweden.

- **Categories:** CAS, Numeric
- **Alignment:** Left
Syntax

\texttt{NLMNISEKw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 12

Range 8–32

\texttt{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the \texttt{d} value is ignored.

Default 2

Range 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input'($12,345.67)',nlmnisek32.2);
y=input'($12,345.67)'dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=;} & -12345.67 \\
\texttt{put y=;} & -12345.67 \\
\hline
\end{tabular}

See Also

Format:

\begin{itemize}
\item \texttt{“NLMNISEKw.d Format” on page 233}
\end{itemize}

\textbf{NLMNISGDw.d Informat}

Reads the monetary format of the international expression for Singapore.

Categories: CAS

Numeric

Alignment: Left
Syntax

\texttt{NLMNISGDw.d}

Syntax Description

\textit{w} 

specifies the width of the output field.

Default 12 

Range 8–32 

\textit{d} 

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2 

Range 0–28 

Example

In the following example, the \texttt{LOCALE=} system option is set to English_UnitedStates.

\begin{verbatim}
x=input'($12,345.67)',nlmnisdg32.2);
y=input'($12,345.67) dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNISGD\texttt{w.d Format}" on page 234

\textbf{NLMNITHBw.d Informat}

Reads the monetary format of the international expression for Thailand.

Categories: CAS 

Numeric 

Alignment: Left
Syntax

**NLMNITHBw.d**

Syntax Description

**w**

specifies the width of the output field.

Default 12

Range 8–32

**d**

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2

Range 0–28

Example

In the following example, the `LOCATE=` system option is set to `English_UnitedStates`.

```r
x=input('($12,345.67)',nlmnithb32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- “NLMNLTHBw.d Informat” on page 622

---

**NLMNITRYw.d Informat**

Reads the monetary format of the international expression for Turkey.

Categories: CAS

Numeric

Alignment: Left
### Syntax

**NLMNITRYw.d**

### Syntax Description

**w**
- Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

**d**
- Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 4
  - Range: 0–28

### Example

In the following example, the LOCATE= system option is set to English UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnitry32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

- **Informat:**
  - “NLMNITRYw.d Informat” on page 623

---

**NLMNITWDw.d Informat**

Reads the monetary format of the international expression for Taiwan.

- **Categories:** CAS, Numeric
- **Alignment:** Left
Syntax

\texttt{NLMNITWDw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\textit{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.

Default: 2

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input'($12,345.67)',nlmnitwd32.2);
y=input'($12,345.67)'dollar32.2);
\end{verbatim}

\begin{verbatim}
Statements Results
----------+----------
put x=; -12345.67
put y=; -12345.67
\end{verbatim}

See Also

\textbf{Format:}

\begin{itemize}
\item \texttt{"NLMNITWDw.d Format" on page 237}
\end{itemize}

\textbf{NLMNIUSDw.d Informat}

Reads the monetary format of the international expression for Puerto Rico and the United States.

Categories: CAS, Numeric

Alignment: Left
Syntax

NLMNIUSDw.d

Syntax Description

w
specifies the width of the output field.

Default  12
Range  8–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default  2
Range  0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input'($12,345.67)',nlmniusd32.2);
y=input'($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLMNIUSDw.d Format" on page 238

NLMNIZARw.d Informat

Reads the monetary format of the international expression for South Africa.

Categories: CAS
Numeric

Alignment: Left
Syntax

Known as

Syntax Description

**w**
specifies the width of the output field.

Default: 12
Range: 8–32

**d**
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default: 2
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)',nlmnizar32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNIZARw.d Format" on page 239

---

**NLMNLAEDw.d Informat**

Reads the monetary format of the local expression for the United Arab Emirates.

**Categories:**
- CAS
- Numeric

**Alignment:**
- Left
Syntax

NLMNLAEDw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 3
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnlaed32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNIAEDw.d Informat” on page 553

NLMNLAUDw.d Informat

Reads the monetary format of the local expression for Australia.

Categories: CAS
            Numeric

Alignment: Left
Syntax

NLMNLAUDw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnlaud32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

“NLMNLAUDw.d Format” on page 241

NLMNLBGNw.d Informat

Reads the monetary format of the local expression for Bulgaria.

Categories: CAS
Numeric

Alignment: Left
Syntax

**NLMNLBGN** *w.d*

### Syntax Description

**w**
- Specifies the width of the output field.
- Default: 12
- Range: 8–32

**d**
- Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
- Default: 2
- Range: 0–28

### Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```
x=input(-12345.67,nlmnlbgn32.2);
y=input(-12345.67,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “**NLMNIBGNw.d Informat**” on page 555

---

**NLMNLBRLw.d Informat**

Reads the monetary format of the local expression for Brazil.

**Categories:**
- CAS
- Numeric

**Alignment:**
- Left
Syntax

**NLMNLBRLw.d**

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 2
  - Range: 0–28

Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```bash
x=input('($12,345.67)',nlmnlbrl32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- "**NLMNIBRLw.d Informat**" on page 556

---

**NLMNLNCADw.d Informat**

Reads the monetary format of the local expression for Canada.

**Categories:**

- CAS
- Numeric

**Alignment:**

- Left
Syntax

**NLMNLCADw.d**

Syntax Description

- **w**: specifies the width of the output field.
  
  Default: 12
  
  Range: 8–32

- **d**: specifies to divide the number by $10^d$. If the data contains decimal points, the **d** value is ignored.
  
  Default: 2
  
  Range: 0–28

Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```plaintext
x=input('($12,345.67)',nlmnlcad32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format**:

- "**NLMNLCA Dw.d Format**" on page 244

---

**NLMNLCHFw.d Informat**

Reads the monetary format of the local expression for Liechtenstein and Switzerland.

Categories: CAS

Numeric

Alignment: Left
Syntax

NLMNLCHFw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

D
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnlchf32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNLCHFw.d Format” on page 245

NLMNLNCNYw.d Informat

Reads the monetary format of the local expression for China.

Categories: CAS, Numeric
Alignment: Left
Syntax

NLMNLCNYw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnlcny32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLMNLCNYw.d Format" on page 246

NLMNLCZKw.d Informat

Reads the monetary format of the local expression for the Czech Republic.

Categories: CAS Numeric

Alignment: Left
Syntax

NLMNLCZK\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default 12

Range 8–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 4

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[x = \text{input}('($12,345.67)', \text{nlnlczyk}32.2);\]
\[y = \text{input}('($12,345.67)', \text{dollar}32.2);\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:
- “NLMNICZKw.d Informat” on page 560

NLMNLDKkw.d Informat

Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.

Categories: CAS, Numeric

Alignment: Left
Syntax

\texttt{NLMNLDKK}w.d

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 12

Range 8–32

\texttt{d}

specifies to divide the number by 10^d. If the data contains decimal points, the \texttt{d} value is ignored.

Default 2

Range 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmnldkk32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=;} & -12345.67 \\
\texttt{put y=;} & -12345.67 \\
\hline
\end{tabular}

See Also

Format:

\begin{itemize}
\item \texttt{"NLMNLDKKw.d Format" on page 248}
\end{itemize}

\textbf{NLMNLEEKw.d Informat}

Reads the monetary format of the local expression for Estonia.

Categories:

\begin{itemize}
\item CAS
\item Numeric
\end{itemize}

Alignment:

Left
Syntax

**NLMNLEEKw.d**

Syntax Description

- **w** specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d** specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 4
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('(12,345.67)',nlnleek32.2);
y=input('(12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- “NLMNIEEKw.d Informat” on page 562
Syntax

**NLMNLEGPw.d**

Syntax Description

**w**

specifies the width of the output field.

Default: 12

Range: 8–32

**d**

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default: 3

Range: 0–28

Example

In the following example, the **locale=** system option is set to **English_UnitedStates**.

```shell
x=input('($12,345.67)',nlmnlegp32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- "**NLMNIEGPw.d Informat**" on page 563
Syntax

\texttt{NLMNLEURw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default: 12

Range: 8–32

\texttt{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 2

Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
    x=input('($12,345.67)',nlmnleur32.2);
    y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- "\texttt{NLMNLEURw.d Format}" on page 251

\textbf{NLMNLGBPw.d Informat}

Reads the monetary format of the local expression for the United Kingdom.

Categories: CAS, Numeric
Syntax

**NLMNLGBP**\(_w.d\)**

Syntax Description

\(w\)

specifies the width of the output field.

- Default: 12
- Range: 8–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- Default: 2
- Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input('($12,345.67)',nlmnlgbp32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "**NLMNLGBPw.d Format**" on page 252

---

**NLMNLHKDw.d Informat**

Reads the monetary format of the local expression for Hong Kong.

Categories:
- CAS
- Numeric
Syntax

NLMNLHKDw.d

Syntax Description

w
  specifies the width of the output field.
  Default 12
  Range 8–32

d
  specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
  Default 2
  Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlhkd32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLHKDw.d Format" on page 253

NLMNLHRKw.d Informat

Reads the monetary format of the local expression for Croatia.

Categories: CAS

Numeric
Syntax

NLMNLHRKw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnlhrk32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Informat:

- "NLMNIHRKw.d Informat" on page 567
Syntax

NLMNLHUFw.d

Syntax Description

w  
specifies the width of the output field.

Default 12
Range 8–32

d  
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$
value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to
English_UnitedStates.

x=input('($12,345.67)',nlmnlhuf32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

| “NLMNIHUFw.d Informat” on page 568 |

NLMNLIDRw.d Informat

Reads the monetary format of the local expression for Indonesia.

Categories: CAS
Numeric
Syntax

NLMNLIDRw.d

Syntax Description

w

specifies the width of the output field.

Default 12

Range 8–32

d

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnlidr32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

NLMNLILSw.d Informat on page 569
Syntax

\texttt{NLMN\textsc{LIL}Sw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

- Default: 12
- Range: 8–32

\textit{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- Default: 4
- Range: 0–28

Example

In the following example, the LOCALE= system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmlils32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=} & -12345.67 \\
\texttt{put y=} & -12345.67 \\
\hline
\end{tabular}

See Also

- Format:
  - "NLMN\textsc{LIL}Sw.d Format" on page 257

\textbf{NLMN\textsc{LINR}w.d Informat}

Reads the monetary format of the local expression for India.

Categories: CAS, Numeric
Syntax

\texttt{NLMNLRw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 12

Range 8–32

\texttt{d}

specifies to divide the number by 10^d. If the data contains decimal points, the \textit{d} value is ignored.

Default 2

Range 0–28

Example

In the following example, the \texttt{locale=} system option is set to \texttt{English_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmnlinr32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
put x=; & -12345.67 \\
put y=; & -12345.67 \\
\hline
\end{tabular}
\end{center}

See Also

**Informat:**

- \texttt{"NLMNIIINRw.d Informat" on page 571}

---

\textbf{NLMNLJPYw.d Informat}

Reads the monetary format of the local expression for Japan.

Categories: CAS

Numeric
Syntax

**NLMNLJPY\(w.d\)**

Syntax Description

\(w\)

- Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

\(d\)

- Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 0
  - Range: 0–28

Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```plaintext
x=input('($12,345.67)',nlmnljpy32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "**NLMNLJPY\(w.d\)** Format" on page 259

**NLMNLKRWw.d Informat**

Reads the monetary format of the local expression for South Korea.

Categories:

- CAS
- Numeric
Syntax

NLMNLRWw.d

Syntax Description

w
 specifies the width of the output field.

Default 12
Range 8–32

d
 specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlnlkrw32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNKRww.d Informat” on page 573

NLMNLLTLw.d Informat

Reads the monetary format of the local expression for Lithuania.

Categories: CAS
Numeric
Syntax

**NLMNLLTL\(w.d\)**

Syntax Description

\(w\)
- Specifies the width of the output field.
- Default: 12
- Range: 8–32

\(d\)
- Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
- Default: 4
- Range: 0–28

Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input('($12,345.67)',nlmnlltl32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**
- "**NLMNILTLw.d Informat**" on page 574

---

**NLMNLLVw.d Informat**

Reads the monetary format of the local expression for Latvia.

**Categories:**
- CAS
- Numeric
Syntax

**NLMNLLVL**$_{w.d}$

Syntax Description

$w$

specifies the width of the output field.

Default 12

Range 8–32

$d$

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default 4

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnllvl32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- “**NLMNILVL**$_w.d$ Informat” on page 575
Syntax

**NLMNLMOPw.d**

**Syntax Description**

- **w** specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d** specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 2
  - Range: 0–28

**Example**

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```
x=input('($12,345.67)',nlmnlmop32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

- **Informat:**
  - "NLMNLMOPw.d Informat" on page 576

**NLMNLMXNw.d Informat**

Reads the monetary format of the local expression for Mexico.

**Categories:**
- CAS
- Numeric
Syntax

**NLMNLMXN**<sub>w.d</sub>

**Syntax Description**

**w**

specifies the width of the output field.

Default: 12

Range: 8–32

**d**

specifies to divide the number by 10<sup>d</sup>. If the data contains decimal points, the <i>d</i> value is ignored.

Default: 2

Range: 0–28

**Example**

In the following example, the LOCALE= system option is set to *English_UnitedStates*.

```plaintext
x=input('($12,345.67)',nlnlnx32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

**Informat:**

- "NLMNIMXNw.d Informat" on page 577

---

**NLMNLMYRw.d Informat**

Reads the monetary format of the local expression for Malaysia.

**Categories:**

- CAS
- Numeric
Syntax

\texttt{NLMNLMYRw.d}

Syntax Description

\textbf{w}

specifies the width of the output field.

Default 12

Range 8–32

\textbf{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 2

Range 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmnlmyr32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLMYRw.d Format" on page 265

NLMNLNOKw.d Informat

Reads the monetary format of the local expression for Norway.

Categories: CAS

Numeric
Syntax

**NLMNLNOK\(w.d\)**

Syntax Description

\(w\)

specifies the width of the output field.

Default 12

Range 8–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{input('($12,345.67)$', nlmnl nok32.2)};
\]

\[
y = \text{input('($12,345.67)$', dollar32.2)};
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNLNOK\(w.d\) Format" on page 266

---

**NLMNLNZD\(w.d\) Informat**

Reads the monetary format of the local expression for New Zealand.

Categories:

- CAS
- Numeric
Syntax

\texttt{NLMNLNZDw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 12

Range 8–32

\texttt{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 2

Range 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to English\_United\_States.

\begin{verbatim}
x=input('($12,345.67)',nlmnlnzd32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

\texttt{"NLMNLNZDw.d Format" on page 267}

NLMNLPLNw.d Informat

Reads the monetary format of the local expression for Poland.

Categories: CAS

Numeric
Syntax

NLMNPLN\textit{w.d}

Syntax Description

\textbf{w}

specifies the width of the output field.

Default: 12  
Range: 8–32

\textbf{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 2  
Range: 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmnlpln32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>-12345.67</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

\textbf{Format:}

\begin{itemize}
  \item \textit{NLMNLPLNw.d Format} on page 268
\end{itemize}
Syntax

NLMNLSEKw.d

Syntax Description

w

specifies the width of the output field.

Default 12
Range 8–32

d

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlrub32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLMNLSEKw.d Informat" on page 269
**Syntax**

\texttt{NLMNLSEKw.d}

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies to divide the number by $10^d$. If the data contains decimal points, the \( d \) value is ignored.
  - Default: 2
  - Range: 0–28

**Example**

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English_UnitedStates}.

```plaintext
x=input('($12,345.67)',nlmnlsek32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

- **Format:**
  - “\texttt{NLMNLSEKw.d Format}” on page 270

**NLMNLSGDw.d Informat**

Reads the monetary format of the local expression for Singapore.
Syntax

\texttt{NLMNLSGDw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

\begin{itemize}
  \item Default: 12
  \item Range: 8–32
\end{itemize}

\textit{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.

\begin{itemize}
  \item Default: 2
  \item Range: 0–28
\end{itemize}

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmnlsgd32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
put x=; & -12345.67 \\
put y=; & -12345.67 \\
\hline
\end{tabular}
\end{table}

See Also

Format:

- “NLMNLSGDw.d Format” on page 271

NLMNLTHBw.d Informat

Reads the monetary format of the local expression for Thailand.
Syntax

**NLMNLTHBw.d**

Syntax Description

* w specifies the width of the output field.
  - Default: 12
  - Range: 8–32

* d specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlthb32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNITHBw.d Informat” on page 585
Syntax

**NLMNLTRYw.d**

Syntax Description

**w**
- Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

**d**
- Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 4
  - Range: 0–28

Example

In the following example, the **LOCALE=** system option is set to **English UnitedStates**.

```plaintext
x=input('($12,345.67)',nlmnltry32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

- **Informat:**
  - "**NLMNITRYw.d Informat**" on page 586

**NLMNLTWDw.d Informat**

Reads the monetary format of the local expression for Taiwan.
Syntax

NLMNLTWDw.d

Syntax Description

w
  specifies the width of the output field.
  Default  12
  Range    8–32

d
  specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
  Default  2
  Range    0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input("($12,345.67)",nlnltwd32.2);
y=input("($12,345.67)",dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLMNLTWDw.d Format" on page 274

NLMNLUSDw.d Informat

Reads the monetary format of the local expression for Puerto Rico, and the United States.
Syntax

**NLMNLUSDw.d**

Syntax Description

- **\( w \)** specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **\( d \)** specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 2
  - Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('(''($12,345.67)',nlmnlusd32.2));
y=input('(''($12,345.67)',dollar32.2));
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- "NLMNLUSDw.d Format" on page 275
Syntax

NLMNLZARw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 12

Range 8–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlzar32.2);
y=input'($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLZARw.d Format” on page 276
NLMNYw.d Informat

Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value.

Categories:
- CAS
- Numeric

Syntax

NLMNYw.d

Syntax Description

- \( w \)
  - specifies the width of the input field.
  - Default: 9
  - Range: 1–32

- \( d \)
  - specifies whether to divide the number by \( 10^d \). If the data contains decimal separators, the \( d \) value is ignored.
  - Default: 0
  - Range: 0–31

Details

The NLMNYw.d informat reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the close parenthesis from the input data.

Comparisons

The NLMNYw.d informat performs processing that is the opposite of the NLMNYIw.d informat.

The NLMNYw.d informat is similar to the DOLLARw.d informat except that the NLMNYw.d informat is locale-specific.

NLMNY returns an error message if you enclose numerical data with apostrophes. The apostrophes specify that the data is character. The following example produces an error message because the numerical value 1 is enclosed in apostrophes.

```plaintext
data;
  x=input('"1"',NLMNY32.);
  put x=;
run;
```
Example

The following examples use the input value of $12,345.67.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>x=input('($12,345.67)',nlmny32.2);</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=input('($12,345.67)',dollar32.2);</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "NLMNYw.d Format" on page 277
- "NLMNYIw.d Format" on page 279

Informat:
- "NLMNYIw.d Informat" on page 629

NLMNYIw.d Informat

Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.

Categories: CAS
Numeric

Syntax

NLMNYIw.d

Syntax Description

\[ w \]

specifies the width of the input field.

Default: 9
Range: 1–32

\[ d \]

specifies whether to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.
Details
The NLMNYI\textsubscript{w.d} informat reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the close parenthesis from the input data.

Comparisons
The NLMNYI\textsubscript{w.d} informat performs processing that is the opposite of the NLMNY\textsubscript{w.d} informat.

Example
The following examples use the input value of 12,345.67.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>x=INPUT('USD12,345.67', NLMNYI32.2);</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=INPUT('$-12,345.67', dollar32.2);</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

- *Formats:*  
  - “NLMNYw.d Format” on page 277  
  - “NLMNYIw.d Format” on page 279

- *Informat:*  
  - “NLMNYw.d Informat” on page 628

**NLNUMw.d Informat**
Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.

Categories: CAS
Syntax

NLNUMw.d

Syntax Description

w
specifies the width of the input field.

Default 6
Range 1–32

d
specifies whether to divide the number by 10d. If the data contains decimal separators, the d value is ignored. If a decimal symbol is not found, the data is regarded as an integer.

Default 0
Range 0–31

Details

The NLNUMw.d) informat reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the close parenthesis from the input data.

NLNUM returns an error message if you enclose numerical data with apostrophes. The apostrophes specify that the data is character. The following example produces an error message because the numeric value 1 is enclosed in apostrophes:

data;
  x=input('1',NLNUM32.);
  put x=;
run;

Comparisons

The NLNUMw.d informat performs processing that is opposite to the NLNUMw.d informat.

Example

The following example uses –1234356.78 as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----+</td>
<td></td>
</tr>
<tr>
<td>NLNUMw.d Informat</td>
<td>631</td>
</tr>
</tbody>
</table>
Statements

```plaintext
options locale=English_UnitedStates;
x=input('-1,234,356.78',nlnum32.2);
put x=;
```

Results

-1234356.78

See Also

Formats:
- “NLNUMw.d Format” on page 281
- “NLMNYlw.d Format” on page 279

Informat:
- “NLNUMIw.d Informat” on page 632

**NLNUMIw.d Informat**

Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.

**Categories:**
- CAS
- Numeric

**Syntax**

`NLNUMI w.d`

**Syntax Description**

`w`

specifies the width of the input field.

- **Default:** 6
- **Range:** 1–32

`d`

specifies to divide the number by $10^d$. If the data contains decimal separators, the $d$ value is ignored.

- **Default:** 0
- **Range:** 0–31
Details

The NLNUMIw.d informat reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the close parenthesis from the input data.

Comparisons

The NLNUMIw.d informat performs processing that is opposite of the NLNUMw.d informat.

Example

The following example uses –1,234,356.78 as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>x=input('-1,234,356.78', nlnumi32.2);</td>
<td>-1234356.78</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

 Formats:

- “NLNUMw.d Format” on page 281
- “NLNUMIw.d Format” on page 282
- “NLNUMw.d Informat” on page 630

NLPCTw.d Informat

Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.

Categories: CAS
            Numeric

Syntax

NLPCTw.d
Syntax Description

\( w \)
specifies the width of the input field.

Default 6
Range 1–32

\( d \)
specifies whether to divide the number by \(10^d\). If the data contains decimal separators, the \( d \) value is ignored.

Default 0
Range 0–31

Details

The NLPCT\( w.d \) informat reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value. It divides the value by 100 and removes any thousands separators, decimal separators, blanks, the percent sign, and the close parenthesis from the input data.

Comparisons

The NLPCT\( w.d \) informat performs processing that is opposite of the NLPCTI\( w.d \) informat. The NLPCT\( w.d \) informat is similar to the PERCENT\( w.d \) informat except that the NLPCT\( w.d \) informat is locale-specific.

NLPCT returns an error message if you enclose numerical data with apostrophes. The apostrophes specify that the data is character. The following example produces an error message because the numerical value 1 is enclosed in apostrophes.

```plaintext
data;
x=input('''1''',NLPCT32.);
put x=;
run;
```

Example

The following example uses –12,345.67% as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates; x=input(''-12,345.67%',nlpct32.2); y=input(''(12,345.67%)'' ,percent32.2); put x=; put y=;</td>
<td>-123.4567</td>
</tr>
</tbody>
</table>
See Also

Formats:
- "NLPCTw.d Format" on page 283
- "NLPCTIw.d Format" on page 285
- "NLPCTIw.d Informat" on page 635

NLPCTIw.d Informat
Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.

Categories: CAS
 Numeric

Syntax

\texttt{NLPCTIw.d}

Syntax Description

\texttt{w}

specifies the width of the input field.

Default 6

Range 1–32

\texttt{d}

specifies whether to divide the number by 10^d. If the data contains decimal separators, the \textit{d} value is ignored.

Default 0

Range 0–31

Details

The NLPCTIw.d informat reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value. It divides the value by 100 and removes any thousands separators, decimal separators, blanks, the percent sign, and the close parentheses from the input data.

Comparisons

The NLPCTIw.d informat performs processing that is opposite of the NLPCTw.d informat.
Example

The following example uses -12,345.67% as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>-123.4567</td>
</tr>
<tr>
<td>x=input('-12,345.67%',nlpct32.2);</td>
<td>-123.4567</td>
</tr>
<tr>
<td>y=input('(12,345.67%)',percent32.2);</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "NLPCTw.d Format" on page 283
- "NLPCTIw.d Format" on page 285

Informat:
- "NLPCTw.d Informat" on page 633

NLSTRMON Informat

Reads the month name in the specified locale and converts it to a numeric value.

Categories: CAS
           Numeric

Syntax

NLSTRMON

Details

The NLSTRMON informat reads the name of the month in the specified locale and converts it to a numeric value. These examples use the English_United States locale:

- The first month (January) = 1
- The second month (February) = 2
- The third month (March) = 3
- The fourth month (April) = 4
- The fifth month (May) = 5
- The sixth month (June) = 6
- The seventh month (July) = 7
- The eight month (August) = 8
- The ninth month (September) = 9
- The 10th month (October) = 10
- The 11th month (November) = 11
- The 12th month (December) = 12

**Example**

This example uses the French_France (fr_fr) locale:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale = fr_fr;</td>
<td>x=1</td>
</tr>
<tr>
<td>data test;</td>
<td>x=janvier</td>
</tr>
<tr>
<td>x = input('janvier',nlstrmon.);</td>
<td>x=5</td>
</tr>
<tr>
<td>put x=;</td>
<td>x=mai</td>
</tr>
<tr>
<td>put x= nlstrmon.;</td>
<td></td>
</tr>
<tr>
<td>x = input('mai',nlstrmon.);</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put x= nlstrmon.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale = en_us;</td>
<td>x=1;</td>
</tr>
<tr>
<td>data test;</td>
<td>x=2;</td>
</tr>
<tr>
<td>x = input(&quot;JANUARY&quot;,nlstrmon.);</td>
<td>x=3;</td>
</tr>
<tr>
<td>put x = ;</td>
<td>x=4;</td>
</tr>
<tr>
<td>x = input(&quot;FEB&quot;,nlstrmon.);</td>
<td></td>
</tr>
<tr>
<td>put x = ;</td>
<td></td>
</tr>
<tr>
<td>x = input(&quot;march&quot;,nlstrmon.);</td>
<td></td>
</tr>
<tr>
<td>put x = ;</td>
<td></td>
</tr>
<tr>
<td>x = input(&quot;apr&quot;,nlstrmon.);</td>
<td></td>
</tr>
<tr>
<td>put x = ;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLTIMAPw. Informat**

Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value.

**Categories:**
- CAS
- Date and Time

**Syntax**

NLTIMAPw.
Syntax Description

\( w \)

- specifies the width of the input field.

Default: 10

Range: 4–200

Example

The following example uses 04:24:43 p.m. as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; 16:24:43</td>
<td></td>
</tr>
<tr>
<td>y=input('04:24:43 PM',nltimap11.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany; 16:24:00</td>
<td></td>
</tr>
<tr>
<td>y=input('16.24 Uhr',nltimap11.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- "NLTIMAPw. Format" on page 294

NLTIMEw. Informat

Reads the time value in the specified locale, and then converts the time value to the local SAS time value.

Categories:

- CAS
- Date and Time

Alias:

- NLTIMAP
Example

The following example uses 16:24:43 as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; y=input('16:24:43',nltime.); put y time.;</td>
<td>16:24:43</td>
</tr>
<tr>
<td>options locale=German_Germany; y=input('16.24 Uhr',nltime.); put y time.;</td>
<td>16:24:00</td>
</tr>
</tbody>
</table>

See Also

Format:
- "NLTIMEw. Format" on page 295

$REVERJw. Informat

Reads character data from right to left and preserves blanks.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$REVERJw.

Syntax Description

\( w \)

specifies the width of the input field.

Default 1 if \( w \) is not specified

Range 1–32767
Comparisons

The $REVERJw. informat is similar to the $REVERSw. informat except that $REVERSw. informat left aligns the result by removing all leading blanks.

Example

The following example uses ABCD as the input value.

input @1 name $reverj7.;

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>#DCBA</td>
</tr>
<tr>
<td>DCBA</td>
<td>#DCBA</td>
</tr>
<tr>
<td>1</td>
<td>#DCBA</td>
</tr>
</tbody>
</table>

1 The character # represents a blank space.

See Also

Informat:

|$REVERSw. Informat$ on page 640

$REVERSw. Informat

Reads character data from right to left, and then left aligns the text.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$REVERSw.

Syntax Description

w specifies the width of the input field.

Default 1 if w is not specified

Range 1–32767
Comparisons

The $REVERSw. informat is similar to the $REVERJw. informat except that $REVERJw. informat preserves all leading and trailing blanks.

Example

The following example uses ABCD as the input value.

```
input @1 name $revers7.;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>ABCD</td>
<td>DCBA###</td>
</tr>
<tr>
<td>ABCD</td>
<td>DCBA###</td>
</tr>
<tr>
<td>1</td>
<td>#</td>
</tr>
</tbody>
</table>

1 The # character represents a blank space.

See Also

Informat:

- "$REVERJw. Informat" on page 639

$UCS2Bw. Informat

Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UCS2Bw.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

- Default: 8
- Range: 2–32767
Comparisons

The $UCS2Bw. informat performs processing that is opposite of the $UCS2BEw. informat. If you are processing data within the same operating environment, then use the $UCS2Xw. informat. If you are processing data from different operating environments, then use the $UCS2Bw. and $UCS2Lw. informats.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('5927'x,$ucs2b.);</td>
<td>x=91e5</td>
</tr>
<tr>
<td>put x=$hex4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

**Formats:**
- "$UCS2Bw. Format" on page 300
- "$UCS2Lw. Format" on page 303
- "$UCS2Xw. Format" on page 305
- "$UTF8Xw. Format" on page 323

**Informats:**
- "$UCS2Lw. Informat" on page 643
- "$UCS2Xw. Informat" on page 646
- "$UTF8Xw. Informat" on page 661

$UCS2Bew. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to big-endian, 16-bit, UCS2, Unicode encoding.

**Category:** Character

**Restriction:** This informat is not supported in a DATA step that runs in CAS.

**Syntax**

$UCS2Bew.
Syntax Description

\[ w \]

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 1–32767

Comparisons

The \$UCS2BE\[w\]. informat performs processing that is opposite of the \$UCS2B\[w\]. informat.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input ('﹫', $ucs2be2.); put ucs2str=$hex4.;</td>
<td>ucs2str=2020</td>
</tr>
</tbody>
</table>

See Also

Formats:
- \"$UCS2Bw. Format\" on page 300
- \"$UCS2BEw. Format\" on page 301

Informat:
- \"$UCS2Bw. Informat\" on page 641

\$UCS2Lw. Informat

Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.
Syntax

\$UCS2Lw.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8

Range 2–32767

Comparisons

The \$UCS2Lw. informat performs processing that is opposite of the \$UCS2LEw. informat. If you are processing data within the same operating environment, then use the \$UCS2Xw. informat. If you are processing data from different operating environments, then use the \$UCS2Bw. and \$UCS2Lw. informats.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('2759'x,$ucs2l.);</td>
<td>x=91e5</td>
</tr>
<tr>
<td>put x=$hex4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- “\$UCS2Bw. Format” on page 300
- “\$UCS2Lw. Format” on page 303
- “\$UCS2Xw. Format” on page 305
- “\$UTF8Xw. Format” on page 323

Informats:

- “\$UCS2Bw. Informat” on page 641
- “\$UCS2Xw. Informat” on page 646
- “\$UTF8Xw. Informat” on page 661
$UCS2LEw. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to little-endian, 16-bit, UCS2, Unicode encoding.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UCS2LEw.

Syntax Description

w specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 2–32767

Comparisons

The $UCS2LEw. informat performs processing that is opposite of the $UCS2Lw. informat.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input (’’, $ ucs2le2.); ucs2str=2759</td>
<td></td>
</tr>
<tr>
<td>put ucs2str=$hex4;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$UCS2Lw. Format" on page 303
- "$UCS2LEw. Format" on page 304
$UCS2Xw. Informat

Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UCS2Xw.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8

Range 2–32767

Comparisons

The $UCS2Xw. informat performs processing that is the opposite of the $UCS2XEw. informat. If you are processing data within the same operating environment, then use the $UCS2Xw. informat. If you are processing data from different operating environments, then use the $UCS2Bw. and $UCS2Lw. informats.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment. This example uses little-endian formatting.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('5927'x,$ucs2x.);</td>
<td>x=91e5</td>
</tr>
<tr>
<td>put x=$hex4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
$UCS2XEw. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to 16-bit, UCS2, Unicode encoding.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UCS2XEw.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8

Range 2-32767

Comparisons

The $UCS2XEw. informat performs processing that is opposite of the $UCS2Xw. informat.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
$\text{UCS4Bw. Informat}$

Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

**Category:** Character

**Restriction:** This informat is not supported in a DATA step that runs in CAS.

**Syntax**

$\text{\$UCS4Bw.}$

**Syntax Description**

- $w$
  
  specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

  - **Default:** 8
  - **Range:** 4–32767

**Comparisons**

If you are processing data within the same operating environment, then use the $\text{\$UCS4Xw.}$ informat. If you are processing data from different operating environments, then use the $\text{\$UCS4Bw.}$ and $\text{\$UCS4Lw.}$ informats.
Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs4str=input(' 大', $ucs4be8.);</td>
<td>ucs4str=00000002000005927</td>
</tr>
</tbody>
</table>

See Also

Format:
- "$UCS4Bw. Format" on page 308

Informats:
- "$UCS4Lw. Informat" on page 649
- "$UCS4Xw. Informat" on page 650

$UCS4Lw. Informat

Reads a character string that is encoded in little-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UCS4Lw

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8

Range 4–32767

Comparisons

If you are processing data within the same operating environment, then use the $UCS4Xw. informat. If you are processing data from different operating environments, then use the $UCS4Bw. and $UCS4Lw. informats.
Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>z=put('.com',$UCS4L16.); put z $hex32.;</td>
<td>2E0000063000006F000006D000000</td>
</tr>
</tbody>
</table>

See Also

Format:
- "$UCS4Lw. Format" on page 311

Informats:
- "$UCS4Bw. Informat" on page 648
- "$UCS4Xw. Informat" on page 650

$UCS4Xw. Informat

Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UCS4Xw.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 8

Range: 4–32767
Comparisons

The $UCS4Xw. informat performs processing that is the opposite of the $UCS4XEw. informat. Use the $UCS4Xw. informat when you are processing data within the same operating environment. Use the $UCS4Bw. and $UCS4Lw. informats when you are processing data from different operating environments.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment. This example uses little-endian formatting.

```
Statements                       Results

    ucs4=put('91e5'x,$ucs4x.);     ucs4=27590000
    sjis=input(ucs4,$ucs4x.);      sjis=91E52020
    put ucs4=$hex8. sjis=$hex8.;   run;
```

See Also

Formats:
- “$UCS2Xw. Format” on page 305
- “$UCS2Bw. Format” on page 300
- “$UCS2Lw. Format” on page 303
- “$UCS4Xw. Format” on page 313
- “$UTF8Xw. Format” on page 323

Informats:
- “$UCS2Bw. Informat” on page 641
- “$UCS2Lw. Informat” on page 643
- “$UTF8Xw. Informat” on page 661

$UCS4XEw. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to 32-bit, UCS4, Unicode encoding.

Category: Character
Restriction: This informat is not supported in a DATA step that runs in CAS.
Syntax

$UCS4XEw.

Syntax Description

w specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8

Range 4–32767

Comparisons

The $UCS4XEw. informat performs processing that is the opposite of the $UCS4Xw. informat.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs4str=input ('ã', $ucs4xe2.);</td>
<td>ucs4str=00005927</td>
</tr>
<tr>
<td>put ucs4str=$hex8;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$UCS4Xw. Format" on page 313
- "$UCS4XEw. Format" on page 315

Informat:

- "$UCS4Xw. Informat" on page 650

$UESCw. Informat

Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.
Syntax

\$UESC\_w.

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 8

Range 1–32767

Details

If the characters are not available on all operating environments (for example, 0–9, a–z, A–Z) they must be represented in UESC representation. The \$UESC\_w. informat can be nested.

Comparisons

The \$UESC\_w. informat performs processing that is the opposite of the \$UESCE\_w. informat.

Example

These examples use the Japanese Shift\_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('¥u5927', $uesc10.); y=input('¥uu5927', $uesc10.); z=input('¥uuu5927', $uesc10.); put x; put y; put z;</td>
<td>¥u5927 ¥uu5927</td>
</tr>
</tbody>
</table>

See Also

Formats:

- "$UESCw. Format" on page 316
- "$UESCEw. Format" on page 317

Informat:
$UESCEw. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UESC representation.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UESCEw.

Syntax Description

w
  specifies the width of the input field.

  Default  8

  Range    1–32767

Details

The $UESCEw. informat can be nested.

Comparisons

The $UESCEw. informat performs processing that is opposite of the $UESCw. informat.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('¥', $uesc10.);</td>
<td>¥u5927</td>
</tr>
<tr>
<td>y=input('¥u5927', $uesc10.);</td>
<td>¥uu5927</td>
</tr>
<tr>
<td>z=input('¥uu5927', $uesc10.);</td>
<td>¥uuu5927</td>
</tr>
<tr>
<td>put x y z;</td>
<td></td>
</tr>
</tbody>
</table>
$UNCRw. Informat

Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.

Category: Character  
Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UNCRw.

Syntax Description

$UNCRw.

w  
   specifies the width of the input field.
   Default  8  
   Range  1–32767

Details

The input string must contain only characters and NCR. Any national characters must be represented in NCR.

Comparisons

The $UNCRw. informat performs processing that is opposite of the $UNCREw. informat.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('大', $uncr10.);</td>
<td>大</td>
</tr>
<tr>
<td>y=input('abc', $uncr10.);</td>
<td>abc</td>
</tr>
<tr>
<td>put X;</td>
<td></td>
</tr>
<tr>
<td>put Y;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

**Formats:**
- “$UNCRw. Format” on page 319
- “$UNCREw. Format” on page 320

**Informat:**
- “$UNCREw. Informat” on page 656

### $UNCREw. Informat

Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.

**Category:** Character

**Restriction:** This informat is not supported in a DATA step that runs in CAS.

**Syntax**

$UNCREw.

**Syntax Description**

**w**
- specifies the width of the input field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1–32767</td>
</tr>
</tbody>
</table>

**Details**

The output string converts to plain characters and NCR. Any national characters convert to NCR.
Comparisons

The $UNCREw. informat performs processing that is the opposite of the $UNCRw. informat.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('￥abc', $uncre12.); put x;</td>
<td>6#22823;abc</td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UNCRw. Format" on page 319
- "$UNCREw. Format" on page 320

Informat:
- "$UNCRw. Informat" on page 655

$UPARENw. Informat

Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UPARENw.

Syntax Description

w

specifies the width of the input field.

Default 8

Range 1–32767
Details
If the SAS session encoding does not have a corresponding Unicode expression, the expression remains in encoding of the current SAS session.

Comparisons
The $UPARENw. informat performs processing that is opposite of the $UPARENEw. informat.

Example
These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('/')&lt;u0061'&gt;',$uparen10.;</td>
<td>a</td>
</tr>
<tr>
<td>w=input('/')&lt;u0062'&gt;',$uparen10.;</td>
<td>b</td>
</tr>
<tr>
<td>x=input('/')&lt;u0063'&gt;',$uparen10.;</td>
<td>c</td>
</tr>
<tr>
<td>y=input('/')&lt;u0033'&gt;',$uparen10.;</td>
<td>3</td>
</tr>
<tr>
<td>z=input('/')&lt;u5927'&gt;',$uparen10.;</td>
<td>\x</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UPARENw. Format" on page 321
- "$UPARENEw. Format" on page 322

Informats:
- "$UPARENEw. Informat" on page 658
- "$UPARENw. Informat" on page 660

$UPARENEw. Informat
Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UPAREN representation.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.
Syntax

\$UPARENEw.

Syntax Description

w

specifies the width of the input field.

Default 8

Range 1–32767

Comparisons

The \$UPARENEw. informat performs processing that is opposite of the \$UPARENw. informat.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('a',$uparen10.);</td>
<td>&lt;u0061&gt;</td>
</tr>
<tr>
<td>w=input('b',$uparene10.);</td>
<td>&lt;u0062&gt;</td>
</tr>
<tr>
<td>x=input('c',$uparene10.);</td>
<td>&lt;u0063&gt;</td>
</tr>
<tr>
<td>y=input('3',$uparene10.);</td>
<td>&lt;u0033&gt;</td>
</tr>
<tr>
<td>z=input('&quot;,&quot;',$uparen10.);</td>
<td>&lt;u5927&gt;</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UPARENw. Format" on page 321
- "$UPARENEw. Format" on page 322

Informats:
- "$UPARENw. Informat" on page 657
- "$UPARENPw. Informat" on page 660
$UPARENPw. Informat

Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session, with national characters that remain in the encoding of the UPAREN representation.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UPARENPw.

Syntax Description

w

specifies the width of the input field.

Default 8

Range 1–32767

Details

If the UPAREN expression contains a national character, whose value is greater than Unicode 0x00ff, the expression remains as a UPAREN expression.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('&lt;u0061&gt;',$uparen10.);</td>
<td>a</td>
</tr>
<tr>
<td>w=input('&lt;u0062&gt;',$uparen10.);</td>
<td>b</td>
</tr>
<tr>
<td>x=input('&lt;u0063&gt;',$uparen10.);</td>
<td>c</td>
</tr>
<tr>
<td>y=input('&lt;u0033&gt;',$uparen10.);</td>
<td>3</td>
</tr>
<tr>
<td>z=input('&lt;u5927&gt;',$uparen10.);</td>
<td>&lt;u5927&gt;</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>
$UTF8Xw. Informat

Reads a character string that is encoded in UTF-8, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$UTF8Xw.

Syntax Description

w

specifies the width of the input field.

Default 8

Range 1–32767

Comparisons

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('e5a4a7' x, $utf8x3.);</td>
<td>x</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>
$VSLOGw. Informat

Reads a character string that is in visual order, and then converts the character string to left-to-right logical order.

Category: BIDI Text Handling

Restriction: This informat is not supported in a DATA step that runs in CAS.

Syntax

$VSLOGw.

Syntax Description

w specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $VSLOGw. informat performs processing that is opposite of the $VSLOGRw. informat.

Example

The following example uses the Hebrew input value of "עברית" (flight).
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = \text{input ('طاس آه',}$vslog12$);$</td>
<td>$\text{flight}$</td>
</tr>
<tr>
<td>\text{put } x;</td>
<td></td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "طاس آه computer."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = \text{input ('طاس آه',}$vslog12$);$</td>
<td>$\text{computer}$</td>
</tr>
<tr>
<td>\text{put } x;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Formats:**
- "$VSLOGRw. Format" on page 327
- "$VSLOGw. Format" on page 325

**Informat:**
- "$VSLOGRw. Informat" on page 663

### $VSLOGRw. Informat

Reads a character string that is in visual order, and then converts the character string to right-to-left logical order.

**Category:**
- BIDI Text Handling

**Restriction:**
- This informat is not supported in a DATA step that runs in CAS.

**Syntax**

\$$VSLOGRw.$$  

**Syntax Description**

- \$w\$
  - specifies the width of the input field.
Comparisons

The $VSLOGRw. informat performs processing that is opposite of the $VSLOGw. informat.

Example

The following example uses the Hebrew input value of "_flight."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('ן ט נ מ ',$vslogr12.);</td>
<td>flight</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "_computer."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('_א ת ז מ א ג נ ','$vslogr12.);</td>
<td>computer</td>
</tr>
<tr>
<td>put x,$vslogr12.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$VSLOGw. Format" on page 325
- "$VSLOGRw. Format" on page 327

Informat:
- "$VSLOGw. Informat" on page 662
YENw.d Informat

Removes embedded yen signs, commas, and decimal points.

Category: Numeric

Restrictions: This informat is not supported in a DATA step that runs in CAS. The YEN informat does not support the UTF-8 encoding. See the NLMNLJGY informat for this functionality.

Syntax

YENw.d

Syntax Description

w

specifies the width of the input field.

Default 1

Range 1–32

d

specifies the power of 10 by which to divide the value.

Requirement  d must be 0 or 2

Tip  If the d is 2, then YENw.d reads a decimal point and two decimal digits. If d is 0, YENw.d reads the value without a decimal part.

Details

The hexadecimal representation of the code for the yen sign character is 5B on EBCDIC systems and 5C on ASCII systems. The monetary character that these codes represent might be different in other countries.

Example

The following example uses yen as the input.

input value yen10.2;

<table>
<thead>
<tr>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>¥1254.71</td>
<td>1254.71</td>
</tr>
</tbody>
</table>
See Also

**Format:**
- “YENw.d Format” on page 333
Macro Functions for NLS
Dictionary of Macro Functions for NLS

Macro Functions by Category

The following table provides brief descriptions of the SAS NLS macro functions. For more information, see the NLS entry for each macro function.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCS</td>
<td>%KCMPRES Macro Function</td>
<td>Compresses multiple blanks and removes leading and trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%QKCMPRES Macro Function (p. 670)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%KINDEX Macro Function</td>
<td>Returns the position of the first character of a string.</td>
</tr>
<tr>
<td></td>
<td>(p. 671)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%KLEFT Macro Function</td>
<td>Left-aligns an argument by removing leading blanks.</td>
</tr>
<tr>
<td></td>
<td>%QKLEFT Macro Function (p. 671)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%KLENGTH Macro Function</td>
<td>Returns the length of a string.</td>
</tr>
<tr>
<td></td>
<td>(p. 672)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%KSCAN Macro Function</td>
<td>Search for a word that is specified by its position in a string.</td>
</tr>
<tr>
<td></td>
<td>%QKSCAN Macro Function (p. 673)</td>
<td></td>
</tr>
</tbody>
</table>
Dictionary

%KCMPRES Macro Function

%QKCMPRES Macro Function

Compresses multiple blanks and removes leading and trailing blanks.

Category: DBCS
Type: NLS macro function

Syntax

%KCMPRES (text | text expression)
%QKCMPRES (text | text expression)

Details

The %KCMPRES and %QKCMPRES macro functions compress multiple blanks and removes leading and trailing blanks. %KCMPRES returns an unquoted result, even if the argument is quoted. %QKCMPRES returns a quoted result.

%QKCMPRES produces a result with the following special characters and mnemonic operators masked, so the macro processor interprets them as text instead of as elements of the macro language:

& % ' " { | ] + - * / < > = ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN
%KINDEX Macro Function

Returns the position of the first character of a string.

Category: DBCS
Type: NLS macro function

Syntax

%KINDEX (source, string)

Required Arguments

source
is a character string or text expression.

string
is a character string or text expression.

Details

The %KINDEX function searches source for the first occurrence of string and returns the position of its first character. If string is not found, the function returns 0.

Example: Locating a Character

The following statements find the first character V in a string:

```sas
%let a=a very long value;
%let b=%kindex(&a,v);
%put V appears at position &b;..
```

When these statements execute, the following line is written to the SAS log:

V appears at position 3.

%KLEFT Macro Function

Left-aligns an argument by removing leading blanks.

Category: DBCS
Requirement: MAUTOSOURCE system option
Syntax

%KLEFT (text | text expression)
%QKLEFT (text | text expression)

Details

The %KLEFT and %QKLEFT macro functions left-align arguments by removing leading blanks. If the argument contains a special character or mnemonic operator, listed here, use QKLEFT.

%KLEFT returns an unquoted result, even if the argument is quoted. %QKLEFT produces a result with the following special characters and mnemonic operators masked so that the macro processor interprets them as text instead of as elements of the macro language:

& % ' " ( ) + - * / < > = ¬ ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN

%KLENGTH Macro Function

Returns the length of a string.

Category: DBCS
Type: NLS macro function

Syntax

%KLENGTH (character string | text expression)

Details

If the argument is a character string, %KLENGTH returns the length of the string. If the argument is a text expression, %KLENGTH returns the length of the resolved value. If the argument has a null value, %KLENGTH returns 0.

Example: Returning String Lengths

The following statements find the lengths of character strings and text expressions:

%let a=Happy;
%let b=Birthday;
%let c=René;
%put The length of &a is %klength(&a).;
%put The length of &b is %klength(&b).;
%put The length of &c is %klength(&c).;
%put The length of &a &b To &c is %klength(&a &b to &c).;

%put with LENGTH macro %length(&c);

When these statements execute, the following is written to the SAS log:

The length of Happy is 5.
The length of Birthday is 8.
The length of René is 4.
The length of Happy Birthday To René is 22.

with LENGTH macro 5

%KSCAN Macro Function

%QKSCAN Macro Function

Search for a word that is specified by its position in a string.

Category: DBCS
Type: NLS macro function

Syntax

\%KSCAN (argument, n<,delimiters >)
\%QKSCAN (argument, n<,delimiters >)

Required Arguments

- **argument**
  - is a character string or a text expression. If argument contains a special character or mnemonic operator, listed here, use %QKSCAN.

- **n**
  - is an integer or a text expression that yields an integer, which specifies the position of the word to return. If n is greater than the number of words in argument, the functions return a null string. If n is negative, %KSCAN examines the character string and selects the word that starts at the end of the string and searches backward.

- **delimiter**
  - specifies a character variable that produces characters that you want %QKSCAN to use as word separators in the character expression.

Details

The %KSCAN and %QKSCAN functions search argument and return the nth word. A word is one or more characters separated by one or more delimiters.

%KSCAN does not mask special characters or mnemonic operators in its results, even when the argument was previously masked by a macro quoting function.

%QKSCAN masks the following special characters and mnemonic operators in its results:

\& % ' " ( ) + − * / < > = ¬ ^ ~ ; , # blank

AND OR NOT EQ NE LE LT GE GT IN
In the `%KSCAN` function, `word` refers to a substring that has all of the following characteristics:

- is bounded on the left by a delimiter or the beginning of the string
- is bounded on the right by a delimiter or the end of the string
- contains no delimiters

A word can have a length of zero if there are delimiters at the beginning or end of the string or if the string contains two or more consecutive delimiters.

If you use the `%KSCAN` function with only two arguments, then the default delimiters depend on whether your computer uses ASCII or EBCDIC characters:

- If your computer uses ASCII characters, then the default delimiters are as follows:
  
  ```
  \('blank' ! $ % & ( ) * + , - . / ; < ^\n\)'
  ```

  In ASCII environments that do not contain the `^` character, the `%KSCAN` function uses the `~` character instead.

- If your computer uses EBCDIC characters, then the default delimiters are as follows:

  ```
  \('blank' ! $ % & ( ) * + , - . / ; < ¬ | ¢\n\)'
  ```

The `%KSCAN` function allows character arguments to be null. Null arguments are treated as character strings with a length of zero. Numeric arguments cannot be null.

Example: Comparing the Actions of `%KSCAN` and `%QKSCAN`

This example illustrates the actions of `%KSCAN` and `%QKSCAN`:

```
%macro a;
   aaaaaa
%mend a;
%macro b;
   bbbbbbb
%mend b;
%macro c;
   ccccccc
%mend c;
%let x=%nrstr(%a*%b*%c);
%put X: &x;
%put The third word in X, with KSCAN: %kscan(&x,3,*);
%put The third word in X, with QKSCAN: %qkscan(&x,3,*);
```

The `%PUT` statement writes these lines to the log:

```
X: %a*%b*%c
The third word in X, with KSCAN: ccccccc
The third word in X, with QKSCAN: %c
```
%KSUBSTR Macro Function

%QKSUBSTR Macro Function

Produce a substring of a character string.

Category: DBCS
Type: NLS macro function

Syntax

%KSUBSTR (argument, position<, length> )
%QKSUBSTR (argument, position<, length> )

Required Arguments

argument
is a character string or a text expression. If argument contains a special character or mnemonic operator, listed here, use %QKSUBSTR.

position
is an integer or an expression (text, logical, or arithmetic) that yields an integer that specifies the position of the first character in the substring. If position is greater than the number of characters in the string, %KSUBSTR and %QKSUBSTR issue a warning message and return a null value.

Optional Argument

length
is an optional integer or an expression (text, logical, or arithmetic) that yields an integer that specifies the number of characters in the substring. If length is greater than the number of characters following position in argument, %KSUBSTR and %QKSUBSTR issue a warning message and return a substring containing the characters from position to the end of the string. By default, %KSUBSTR and %QKSUBSTR produce a string containing the characters from position to the end of the character string.

Details

The %KSUBSTR and %QKSUBSTR functions produce a substring of argument, which begins at position and continues for the number of characters in length.

%KSUBSTR does not mask special characters or mnemonic operators in its result.

%QSUBSTR masks the following special characters and mnemonic operators:

& % ' " ( ) − * / < > = ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN
Examples:

Example 1: Limiting a Fileref to Eight Characters

The macro MAKEFREF uses %KSUBSTR to assign the first eight characters of a parameter as a fileref, in case a user assigns one that is longer:

```sas
%macro makefref(fileref,file);
  %if %klength(&fileref) gt 8 %then
    %let fileref = %ksubstr(&fileref,1,8);
  filename &fileref "&file";
%mend makefref;
%makefref(humanresource,/dept/humanresource/report96)
```

SAS reads the following statement:

```sas
FILENAME HUMANRES "/dept/humanresource/report96";
```

Example 2: Storing a Long Macro Variable Value in Segments

The macro SEPMSG separates the value of the macro variable MSG into 40-character units and stores each unit in a separate variable:

```sas
%macro sepmsg(msg);
  %let i=1;
  %let start=1;
  %if %length(&msg)>40 %then
    %do;
      %do %until(%klength(&&msg&i)<40);
        %let msg&i=%qksubstr(&msg,&start,40);
        %put Message &i is: &&msg&i;
        %let i=%eval(&i+1);
        %let start=%eval(&start+40);
        %let msg&i=%qksubstr(&msg,&start);
      %end;
      %put Message &i is: &&msg&i;
    %end;
  %else %put No subdivision was needed.;
%mend sepmsg;
%sepmsg(%nrstr(A character operand was found in the %EVAL function or %IF condition where a numeric operand is required. A character operand was found in the %EVAL function or %IF condition where a numeric operand is required.));
```

When this program executes, these lines are written to the SAS log:

```
Message 1 is: A character operand was found in the %EV
Message 2 is: AL function or %IF condition where a nu
Message 3 is: meric operand is required. A character
Message 4 is: operand was found in the %EVAL function or %IF condition where a numeric operand is required.)));
```

Example 3: Comparing the Actions of %KSUBSTR and %QKSUBSTR

%KSUBSTR produces a resolved result because it does not mask special characters and mnemonic operators in the C language before processing it:
%let a=one;
%let b=two;
%let c=%nrstr(&a &b);
%put C: &c;
%put With KSUBSTR: %ksubstr(&c,1,2);
%put With QKSUBSTR: %qKsubstr(&c,1,2);

When these statements execute, these lines are written to the SAS log:

C: &a &b
With KSUBSTR: one
With QKSUBSTR: &a

%KUPCASE Macro Function

%QKUPCASE Macro Function

Convert values to uppercase.

Category:  DBCS
Type:      NLS macro function

Syntax

%KUPCASE (character string | text expression)
%QKUPCASE (character string | text expression)

Details

The %KUPCASE and %QKUPCASE functions convert lowercase characters in the
argument to uppercase. %KUPCASE does not mask special characters or
mnemonic operators in its results.

If the argument contains a special character or mnemonic operator, listed here, use
%QKUPCASE. %QKUPCASE masks the following special characters and
mnemonic operators in its results:
& % ' " ( ) + - * / < > = ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN

%KUPCASE and %QKUPCASE are useful in comparing values because the macro
facility does not automatically convert lowercase characters to uppercase before
comparing them.
Examples:

Example 1: Capitalizing a Value to Be Compared
In this example, the macro RUNREPT compares a value input for the macro variable MONTH to the string DEC. If the uppercase value of the response is DEC, then PROC FSVIEW runs on the data set REPORTS.ENDYEAR. Otherwise, PROC FSVVIEW runs on the data set with the name of the month in the REPORTS data library.

```
%macro runrept(month);
  %if %kupcase(&month)=DEC %then
    %str(proc fsview data=reports.endyear; run;);
  %else %str(proc fsview data=reports.&month; run;);
%mend runrept;
```
You can invoke the macro in any of these ways to satisfy the %IF condition:
%runrept(DEC)
%runrept(Dec)
%runrept(dec)

Example 2: Comparing %KUPCASE and %QKUPCASE
These statements show the results produced by %KUPCASE and %QKUPCASE:
```
%let a=begin;
%let b=%nrstr(&a);
%put KUPCASE produces: %kupcase(&b);
%put QKUPCASE produces: %qkupcase(&b);
```
When these statements execute, the following is written to the SAS log:
KUPCASE produces: BEGIN
QKUPCASE produces: &A

%VALIDCHS Macro Function
Validates the character(s) encoding compatibility for data set variables.

**Restriction:**
This macro should not be used when the encoding value equals DEFAULT.

**Syntax**

```
%VALIDCHS(dsnn=dataset_name,
            libnn=libref_name<ENCODING=encoding_name,><COMPATIBLE=encoding_name>)
```

**Required Arguments**
- **dsn**=dataset_name
  name of the data set to be verified.
- **lib**=libref_name
  name of the SAS library to be verified.
Optional Arguments

**ENCODING=encoding_name**
optional encoding name that the %VALIDCHS macro function uses to validate that the characters in the data set are valid for the specified encoding. If the ENCODING= option is not specified, the encoding of the data set is used.

**COMPATIBLE=encoding_name**
optional encoding that the %VALIDCHS macro function uses to verify the compatibility of the data in the data set with the specified encoding. It helps determine whether you can successfully transfer the data set between a server and a client where the encoding is different. Incompatible characters can cause data transfer issues.

Details

Basics
The %VALIDCHS macro function scans the data set and inspects the characters in the data set's character variables to determine whether they are valid for the specified encoding. If an ENCODING= option is specified, the %VALIDCHS macro function validates based on the ENCODING= value. If the ENCODING= option is not specified, then %VALIDCHS uses the encoding of the data set. If the COMPATIBLE= option is specified, %VALIDCHS validates the contents of the character data in the encoding of the data set (or the encoding of the ENCODING= option). Then it verifies the compatibility based on the encoding that is specified by the COMPATIBLE= option. For example, a data set with a euc-cn encoding, Chinese characters, and COMPATIBLE=wlatin1 is specified. %VALIDCHS verifies that the characters are valid for the euc-cn encoding. Then %VALIDCHS verifies the compatibility of the characters to see whether they are supported by the wlatin1 encoding. If %VALIDCHS detects invalid character(s) in the data set, it displays a message and continues validation.

Automatic Macro Variable

_validchars_rc is an automatic macro variable that is generated by %VALIDCHS. It indicates the test status. Here is an example:

```
2:  No invalid characters are found but data will cause the truncation
1:  No invalid characters are found
0:  Invalid characters are found
-1:  Invalid characters are found and also data will cause the truncation
-2:  General error
```

Examples:

**Example 1: Simple Validation of the Character Data in the Data Set**

%VALIDCHS uses the encoding of the input data set.

```sas
%VALIDCHS(dsnm=sashelp.class);
%put &_validchars_rc;
```

The encoding of the data set is us-ascii. The %VALIDCHS macro function finds that all the character data is valid in us-ascii so that it returns following message in the log:
Example 2: Validate the Character Data in the Data Set in the Specified Encoding

%VALIDCHS uses the specified encoding to validate these characters:

```sas
%VALIDCHS(dsnm=sashelp.class, encoding=utf-8);
%put &_validchars_rc;
```

These messages are written to the SAS log:

```
NOTE: All characters in the dataset: sashelp.class (encoding:utf-8) are valid.
0
```

Example 3: Validate the Character Data and Check the Compatibility

```sas
%VALIDCHS(dsnm=sashelp.class, encoding=euc-cn, compatible=wlatin1);
%put &_validchars_rc;
```

These messages are written to the SAS log:

```
NOTE: All characters in the dataset: sashelp.class (encoding:euc-cn) are compatible to wlatin1 encoding.
0
```

Example 4: Validate the Character Data and Check the Compatibility

The following data set has the wlatin1 encoding and a national character:

```sas
data mydata;
  length ch $1.;
  ch = '€';
run;
%VALIDCHS(dsnm=mydata, encoding=wlatin1, compatible=utf-8);
%put &_validchars_rc;
```

These messages are written to the SAS log:

```
WARNING: Found possible truncation in the variable: ch at observation number: 1
ERROR: Detected the issue in the dataset: mydata (encoding:wlatin1) when validating the compatibility to ::utf-8 encoding.
2
```
PART 8

System Options for NLS

Chapter 16

Dictionary of System Options for NLS

683
# Dictionary of System Options for NLS

## System Option Entries by Category

The language control category of SAS system options are affected by NLS. The following table provides brief descriptions of the SAS system options. For more detailed descriptions, see the dictionary entry for each SAS system option:

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment Control:</td>
<td>DATESTYLE= System Option (p. 686)</td>
<td>Specifies the sequence of month, day, and year when ANYDTDTE, ANYYDITDM, or ANYYDTTME informat data is ambiguous.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Language Control</td>
<td>DBCS System Option: UNIX, Windows, and z/OS (p. 687)</td>
<td>Recognizes double-byte character sets (DBCS).</td>
</tr>
<tr>
<td></td>
<td>DBCSLANG System Option: UNIX, Windows, and z/OS (p. 688)</td>
<td>Specifies a double-byte character set (DBCS) language.</td>
</tr>
<tr>
<td></td>
<td>DBCSTYPE System Option: UNIX, Windows, and z/OS (p. 689)</td>
<td>Specifies the encoding method to use for a double-byte character set (DBCS).</td>
</tr>
<tr>
<td></td>
<td>DFLANG= System Option: UNIX, Windows, and z/OS (p. 691)</td>
<td>Specifies the language for international date informats and formats.</td>
</tr>
<tr>
<td></td>
<td>ENCODING System Option: UNIX, Windows, and z/OS (p. 694)</td>
<td>Specifies the default character-set encoding for the SAS session.</td>
</tr>
<tr>
<td></td>
<td>FSDBTYPE System Option: UNIX (p. 695)</td>
<td>Specifies a full-screen double-byte character set (DBCS) encoding method.</td>
</tr>
<tr>
<td></td>
<td>FSIMM System Option: UNIX (p. 696)</td>
<td>Specifies input method modules (IMMs) for full-screen double-byte character set (DBCS).</td>
</tr>
<tr>
<td></td>
<td>FSIMMOPT System Option: UNIX (p. 697)</td>
<td>Specifies options for input method modules (IMMs) that are used with a full-screen double-byte character set (DBCS).</td>
</tr>
<tr>
<td></td>
<td>LOCALE System Option (p. 698)</td>
<td>Specifies a set of attributes in a SAS session that reflect the language, local conventions, and culture for a geographical region.</td>
</tr>
<tr>
<td></td>
<td>LOCALEDATA System Option: UNIX, Windows, and z/OS (p. 700)</td>
<td>Specifies the source database for the locale information.</td>
</tr>
<tr>
<td></td>
<td>LOGLANGENG System Option (p. 701)</td>
<td>Specifies using the English language for SAS log message text when the LOCALE option is set after start-up.</td>
</tr>
<tr>
<td></td>
<td>MAPEBCDIC2ASCII= System Option (p. 704)</td>
<td>Specifies a translation table that SAS uses to transcode from EBCDIC to ASCII and from ASCII to EBCDIC.</td>
</tr>
<tr>
<td></td>
<td>ODSLANGCHG System Option (p. 706)</td>
<td>Determines whether the language of the text of the ODS output can be changed.</td>
</tr>
<tr>
<td></td>
<td>TIMEZONE= System Option (p. 709)</td>
<td>Specifies the user local time zone.</td>
</tr>
<tr>
<td></td>
<td>TRANTAB= System Option (p. 711)</td>
<td>Specifies the translation tables that are used by various parts of SAS.</td>
</tr>
<tr>
<td></td>
<td>URLENCODING= System Option (p. 718)</td>
<td>Specifies whether the argument to the URLENCODE function and to the URLDECODE function is interpreted using the SAS session encoding or UTF-8 encoding.</td>
</tr>
</tbody>
</table>
**Category** | **Language Elements** | **Description**
---|---|---
Files: External Files | BOMFILE System Option (p. 685) | Specifies whether to write the byte-order mark (BOM) prefix on Unicode-encoded external files.
Files: SAS Files | RSASIOTRANSERROR System Option (p. 707) | Displays a transcoding error when invalid data is read from a remote application.
 | VALIDMEMNAME= System Option (p. 713) | Specifies the rules for naming SAS data sets, SAS data views, and item stores.
 | VALIDVARNAME= System Option (p. 716) | Specifies the rules for valid SAS variable names that can be created and processed during a SAS session.
Input Control: Data Processing | DATESTYLE= System Option (p. 686) | Specifies the sequence of month, day, and year when ANYDTDTE, ANYDTDTM, or ANYDTTME informat data is ambiguous.
Language Control | LSWLANG System Option (p. 703) | Specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation.
Sort: Procedure Options | SORTSEQ= System Option: UNIX, Windows, and z/OS (p. 708) | Specifies a language-specific collating sequence for the SORT and SQL procedures to use in the current SAS session.

---

**Dictionary**

**BOMFILE System Option**

Specifies whether to write the byte-order mark (BOM) prefix on Unicode-encoded external files.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: External Files

PROC OPTIONS GROUP=

---

**Syntax**

**BOMFILE | NOBOMFILE**

**Syntax Description**

**BOMFILE**

Specifies to write a byte-order mark (BOM) prefix when a Unicode-encoded file is written to an external file.
NOBOMFILE
Specifies not to write a BOM prefix when a Unicode-encoded file is written to an external file.

Details
The BOMFILE system option does not apply when a Unicode-encoded external file is read.
A BOM is a signature at the beginning of a Unicode data stream. The size of the BOM varies depending on the encoding.

DATESTYLE= System Option
Specifies the sequence of month, day, and year when ANYDTDTE, ANYDTDTM, or ANYTTME informat data is ambiguous.

Valid in:
SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable (UNIX only), SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable (Linux only)

Categories:
Environment Control: Language Control
Input Control: Data Processing

PROC OPTIONS GROUP=
INPUTCONTROL
LANGUAGECONTROL

Default:
The default value is determined by the value of the LOCALE= system option.

Note:
This option cannot be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

Syntax

DATESTYLE= MDY | YMD | DMY | LOCALE

Syntax Description

MDY
specifies that SAS set the order as month, day, year.

YMD
specifies that SAS set the order as year, month, day.

DMY
specifies that SAS set the order as day, month, year.

LOCALE
specifies that SAS set the order based on the value that corresponds to the LOCALE= system option value and is one of the following: MDY | YMD | DMY.
Details

System option DATESTYLE= identifies the order of month, day, and year. The default value is LOCALE. The default LOCALE system option value is English. Therefore, the default DATESTYLE order is MDY.

To get the default settings for each locale option value, see Locale Values on page 811.

See Also

Informats:

- “ANYDTDTEw. Informat” in SAS Formats and Informats: Reference
- “ANYDTEw. Informat” in SAS Formats and Informats: Reference
- “ANYDTTMEw. Informat” in SAS Formats and Informats: Reference

System Options:

- “LOCALE System Option” on page 698

DBCS System Option: UNIX, Windows, and z/OS

Recognizes double-byte character sets (DBCS).

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS LANGUAGECONTROL
Default: NODBCS
UNIX specifics: Also valid in SASV9_OPTIONS environment variable

Syntax

-DBCS | -NODBCS (UNIX and Windows)
DBCS | NODBCS ( z/OS)

Required Arguments

**DBCS**
recognizes double-byte character sets (DBCS) for encoding values. DBCS encodings are used to support East Asian languages.

**NODBCS**
does not recognize a DBCS for encoding values. Instead, a single-byte character set (SBCS) is used for encoding values. A single byte is used to represent each character in the character set.
Details

The DBCS system option is used for supporting languages from East Asian
countries such as Chinese, Japanese, Korean, and Taiwanese.

See Also

Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 45
- “DBCS Values for a SAS Session” on page 827
- Chapter 23, “Encoding Values in SAS Language Elements,” on page 829

System Options:
- “DBCSLANG System Option: UNIX, Windows, and z/OS” on page 688
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 689

DBCSLANG System Option: UNIX, Windows, and z/OS

Specifies a double-byte character set (DBCS) language.

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS GROUP= LANGUAGECONTROL
Default: none
UNIX specifics: Also valid in SASV9_OPTIONS environment variable

Syntax

-DBCSLANG language (UNIX and Windows)

DBCSLANG = language ( z/OS)

Required Argument

language depends on the operating environment. The following table contains valid
language values:

Table 16.1 Supported DBCS Languages According to Operating Environment

<table>
<thead>
<tr>
<th>Language</th>
<th>z/OS</th>
<th>UNIX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHINESE (simplified)</td>
<td>yes†</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
### DBCS Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>z/OS</th>
<th>UNIX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAPANESE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>KOREAN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TAIWANESE (traditional)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NONE</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

1 For z/OS only, HANGUL is a valid alias for KOREAN and HANZI is a valid alias for CHINESE.

### Details

The proper setting for the DBCSLANG system option depends on which setting is used for the DBCSTYPE system option. Some of the settings of DBCSTYPE support all of the DBCSLANG languages. Other settings of DBCSTYPE support only Japanese.

CHINESE specifies the language used in the People's Republic of China, which is known as simplified Chinese. TAIWANESE specifies the Chinese language used in Taiwan, which is known as traditional Chinese.

### See Also
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 45
- “DBCS Values for a SAS Session” on page 827
- Chapter 23, “Encoding Values in SAS Language Elements,” on page 829

### System Options:
- “DBCS System Option: UNIX, Windows, and z/OS” on page 687
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 689

---

**DBCSTYPE System Option: UNIX, Windows, and z/OS**

Specifies the encoding method to use for a double-byte character set (DBCS).

**Valid in:** configuration file, SAS invocation

**Category:** Environment Control: Language Control

**PROC OPTIONS GROUP=LANGUAGECONTROL**

**z/OS specifics:** IBM
UNIX specifics:
Depends on the specific machine
Also valid in SASV9_OPTIONS environment variable

Windows specifics:
PCMS

Syntax
-DBCSTYPE encoding-method (UNIX and Windows)

DBCSTYPE = encoding-method ( z/OS)

Required Argument
encoding-method
specifies the method that is used to encode a double-byte character set (DBCS). Valid values for encoding-method depend on the standard that the computer hardware manufacturer applies to the operating environment.

Details
DBCS encoding methods vary according to the computer hardware manufacturer and the standards organization.

The DBCSLANG= system option specifies the language that the encoding method is applied to. You should specify DBCSTYPE= only if you also specify the DBCS and DBCSLANG= system options.

z/OS DBCSTYPE= supports the DBCSTYPE= value of IBM.

Comparisons

Table 16.2  DBCS Encoding Methods for z/OS

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>IBM EBCDIC encoding method</td>
</tr>
<tr>
<td>PCIBM</td>
<td>IBM PC encoding method</td>
</tr>
</tbody>
</table>

Table 16.3  DBCS Encoding Methods for UNIX

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>DEC encoding method</td>
</tr>
<tr>
<td>EUC</td>
<td>Extended UNIX Code encoding method</td>
</tr>
<tr>
<td>HP15</td>
<td>Hewlett Packard encoding method</td>
</tr>
<tr>
<td>PCIBM</td>
<td>IBM PC encoding method</td>
</tr>
<tr>
<td>DBCSTYPE= Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>PCMS</td>
<td>Microsoft PC encoding method</td>
</tr>
<tr>
<td>SJIS</td>
<td>Shift-JIS encoding method for the Japanese language only</td>
</tr>
<tr>
<td>NONE</td>
<td>Disables DBCS processing</td>
</tr>
</tbody>
</table>

Table 16.4  DBCS Encoding Methods for Windows

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCMS</td>
<td>Microsoft PC encoding method</td>
</tr>
<tr>
<td>PCIBM</td>
<td>IBM PC encoding method</td>
</tr>
<tr>
<td>WINDOWS</td>
<td>Alias for PCMS</td>
</tr>
<tr>
<td>SJIS</td>
<td>Shift-JIS encoding method for the Japanese language only</td>
</tr>
</tbody>
</table>

See Also

Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 45
- “DBCS Values for a SAS Session” on page 827
- Chapter 23, “Encoding Values in SAS Language Elements,” on page 829

System Options:
- “DBCS System Option: UNIX, Windows, and z/OS” on page 687
- “DBCSLANG System Option: UNIX, Windows, and z/OS” on page 688

DFLANG= System Option: UNIX, Windows, and z/OS

Specifies the language for international date informats and formats.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category: Environment Control: Language Control
PROC OPTIONS GROUP=
Default: English
Syntax

**DFLANG='language', locale**

Syntax Description

*language*

specifies the language that is used for international date informats and formats.

These languages are valid values for *language*:

- Afrikaans
- Catalan
- Croatian
- Czech
- Danish
- Dutch
- English
- Finnish
- French
- German
- Hungarian
- Italian
- Japanese
- Macedonian
- Norwegian
- Polish
- Portuguese
- Russian
- Slovenian
- Spanish
- Swedish
- Swiss_French
- Swiss_German

**locale**

the locale that is specified with the locale system option becomes the active locale.

Details

You can change the value of the DFLANG system option during a SAS session, but you can use only one language at a time. The values for *language* are not case-sensitive.
When you specify `DFLANG=locale`, the locale that is specified in the system option of the `LOCALE` statement becomes the active locale. The locale or language must be supported by the `DFLANG` system option.

In the following example, the international date informats and formats would be German. The posix name for the German locale is `de_DE`. The German locale is supported by the `DFLANG` system option.

```
option locale=de_DE; /* German locale */
option DFLANG=locale;
```

In the following example, the international date informats and formats would be English. Maltese is not supported by `DFLANG`, so the default locale is English.

```
option locale=mt_MT; /* Maltese locale */
option DFLANG=locale;
```

When you specify `DFLANG=locale`, the output of the date format is displayed in the locale that is specified with the `LOCALE` system option. To control the date format in the output, the `DFLANG` locale uses the value based on the `LOCALE` system option that has been set at start-up. If `DFLANG` is set to a valid language, then the date format in the output is English by default. In the following example, the locale is set to French:

```
Sas.exe –locale French
   Proc print data=sashelp.class ; run ;
```

```
mercredi 09 mars 2011 14 h 25
```

If you set `DFLANG=JAPANESE`, then `DFLANG` behaves the same way as `DFLANG=LOCALE`.

The value `JAPANESE` is supported to allow customers running the Japanese image to see the datetime stamp in the SAS Log and the Listing formatted for Japanese.

- If `DFLANG` is not set at start up, the option value is set by the `LOCALE` option.
- The default format of the datetime stamp of the SAS log is English. If `DFLANG` is set to `LOCALE` or `JAPANESE`, the datetime stamp in the SAS Log is formatted according to the current SAS `LOCALE`.
- The default format of the datetime stamp of the SAS listing is English. If `DFLANG` is set to `LOCALE` or `JAPANESE`, the datetime stamp in each page of the output sent to the SAS Listing or ODS destinations is formatted according to the SAS `LOCALE` that was in effect immediately after `DFLANG=LOCALE` is set.
- Changes to the `LOCALE` option do not impact how the datetime stamp is formatted unless the `DTRESET` option is enabled. Then the datetime stamp is formatted according to the current SAS `LOCALE`.

You can use the value `JAPANESE` to process the Japanese image to see the datetime stamp in the SAS log and the listing formatted for Japanese. The value `LOCALE` is also supported.

See Also

* DTRESET System Option*
ENCODING System Option: UNIX, Windows, and z/OS

Specifies the default character-set encoding for the SAS session.

Valid in: configuration file, SAS invocation

Category: Environment Control: Language Control

PROC OPTIONS GROUP=

UNIX specifics: latin1
Windows specifics: wlatin1
z/OS specifics: OPEN_ED-1047

Syntax

-ENCODING= ASCIIANY | EBCDICANY | encoding-value (UNIX and Windows)
ENCODING= encoding-value (UNIX, Windows, and z/OS)

Required Arguments

ASCIIANY

Transcoding normally occurs when SAS detects that the session encoding and data set encoding are different. ASCIIANY enables you to create a data set that SAS will not transcode if the SAS session that accesses the data set has a session that encoding value of ASCII. If you transfer the data set to a machine that uses EBCDIC encoding, transcoding occurs.

Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

EBCDICANY

is valid only for z/OS. Transcoding normally occurs when SAS detects that the session encoding and the data set encoding are different. EBCDICANY enables you to create a data set that SAS does not transcode if the SAS session accessing the data set has a session encoding value of EBCDIC. If you transfer the data set to a machine that uses ASCII encoding, transcoding occurs.

encoding-value

For valid values for all operating environments, see Chapter 24, “Encoding Values for a SAS Session,” on page 841.

Details

A character-set encoding is a set of characters that have been mapped to numeric values called code points.

The encoding for a SAS session is determined by the values of the ENCODING=, LOCALE=, DBCSTYPE=, and DBCSLANG= system options as follows:
If the ENCODING option is not specified, the value of Encoding is determined by the value of LOCALE and the operating system where SAS is running. Also, if LOCALE is not set, the default LOCALE is en_US.

If both LOCALE= and ENCODING= are specified, the session encoding is the value that is specified by the ENCODING= option.

If LOCALE= is specified and ENCODING= is not specified, SAS infers the appropriate encoding value from the LOCALE= value.

If the DBCS option is set, the values for the DBCSLANG= and DBCSTYPE= system options determine the ENCODING= and LOCALE= values.

See Also

Conceptual Information:
- “Overview of Locale Concepts for NLS” on page 5
- Conceptual discussion about “Overview: Encoding for NLS” on page 9
- Conceptual discussion about “Overview of Transcoding” on page 31
- Table 21.1 on page 811
- Chapter 22, “SAS System Options for Processing DBCS Data,” on page 827
- Chapter 23, “Encoding Values in SAS Language Elements,” on page 829

FSDBTYPE System Option: UNIX
Specifies a full-screen double-byte character set (DBCS) encoding method.

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment Control: Language Control
PROC OPTIONS GROUP=
Default: DEFAULT
UNIX specifics: all

Syntax
-FSDBTYPE encoding-method

Details
The FSDBTYPE= system option specifies the encoding method that is appropriate for a full-screen DBCS enabling method. Full-screen DBCS encoding methods vary according to the computer hardware manufacturer and the standards organization.
### Table 16.5  Full-Screen DBCS Encoding Methods

<table>
<thead>
<tr>
<th>FSDBTYPE= Encoding Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dec</td>
<td>Digital Equipment Corporation encoding method</td>
</tr>
<tr>
<td>euc</td>
<td>Extended UNIX encoding method</td>
</tr>
<tr>
<td>hp15</td>
<td>HP-UX encoding method</td>
</tr>
<tr>
<td>jis7</td>
<td>7-bit Shift-JIS encoding method used in an X windows environment for the Japanese language only</td>
</tr>
<tr>
<td>pcibm</td>
<td>IBM PC encoding method</td>
</tr>
<tr>
<td>sjis</td>
<td>Shift-JIS encoding method for the Japanese language only</td>
</tr>
<tr>
<td>default</td>
<td>default method that is used by the specific host</td>
</tr>
</tbody>
</table>

**See Also**

Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 45
- “DBCS Values for a SAS Session” on page 827
- Chapter 23, “Encoding Values in SAS Language Elements,” on page 829

---

### FSIMM System Option: UNIX

Specifies input method modules (IMMs) for full-screen double-byte character set (DBCS).

- **Valid in:** configuration file, SAS invocation, SASV9_OPTIONS environment variable
- **Category:** Environment Control: Language Control
- **PROC OPTIONS GROUP=** LANGUAGECONTROL
- **Default:** none
- **UNIX specifics:** all

**Syntax**

```
-FSIMM fsdevice_name=IMM-name1<, fsdevice_name=IMM-name2> ...
```
Details

You can specify the following values for IMM-name:

TTY | SASWUJT
   provides an interface for /dev/tty. This IMM enables you to enter DBCS strings through a terminal emulator that has DBCS input capability.

PIPE | SASWUJP
   provides a pipe interface. This interface forks the DBCS input server process. The default server name is saswujms, which uses the vendor-supplied MOTIF toolkit.

For example, to use the PIPE input method module for X11 drivers, you would specify this code:

-FSIMM X11=PIPE

Note: The server is specified by using the FSIMMOPT option.

See Also

Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 45

System Option:
- “FSIMMOPT System Option: UNIX” on page 697

FSIMMOPT System Option: UNIX

Specifies options for input method modules (IMMs) that are used with a full-screen double-byte character set (DBCS).

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment Control: Language Control
PROC OPTIONS GROUP=
Default: none
UNIX specifics: all

Syntax

-FSIMMOPT fullscreen-IMM:IMM-option
Details

The FSIMMOPC system option specifies an option for each full-screen IMM (input method module). You can specify only one FSIMMOPC option for each IMM. If you specify multiple FSIMMOPC options for the same IMM, only the last specification is used.


For example, you can use the FSIMMOPC option to specify the name of the server, MOTIF, to be used for the PIPE IMM:

\[-fsimmopt PIPE:MOTIF\]

See Also

**Conceptual Information:**

- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 45

**System Option:**

- “FSIMM System Option: UNIX” on page 696

---

**LOCALE System Option**

Specifies a set of attributes in a SAS session that reflect the language, local conventions, and culture for a geographical region.

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment Control: Language Control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**Default:** English_UnitedStates

**UNIX specifics:** Also valid in SASV9_OPTIONS environment variable

**Syntax**

- **-LOCALE locale-name** (UNIX and Windows)

  **LOCALE=locale-name** (UNIX, Windows, and z/OS)

**Required Argument**

**locale-name**

For a complete list of locale values (SAS names and POSIX names), see “LOCALE= Values for PAPERSIZE and DFLANG, Options” on page 811.
Details

The LOCALE= system option is used to specify the locale, which reflects the local conventions, language, and culture of a geographical region.

If the value of the LOCALE= system option is not compatible with the value of the ENCODING= system option, the character-set encoding is determined by the value of the ENCODING= system option.

If the DBCS= system option is active, the values of the DBCSTYPE= and DBCSLANG= system options determine the locale and character-set encoding.

When you set a value for LOCALE=, the value of the following system options is modified unless explicit values have been specified:

ENCODING=
   The locale that you set has a common encoding value that is used most often in the operating environment where SAS runs. If the ENCODING= option is not set explicitly in a config file or on the command line, SAS uses the ENCODING that is default for the LOCALE and operating system. The LOCALE might be set explicitly or can default. When the ENCODING= system option is set, the TRANTAB= system option is also set.

DATESTYLE=
   When LOCALE= is set, the DATESTYLE= system option uses the value that corresponds to the chosen locale.

DFLANG=
   When LOCALE= is set, the DFLANG= system option is set to a value that corresponds to the chosen locale.

PAPERSIZE=
   When LOCALE= is set, the PAPERSIZE= system option is set to a value that corresponds to the chosen locale and the ODS printer is set to the preferred unit of measurement, inches or centimeters, for that locale.

CAUTION
Under the Windows operating systems only: The LOCALE= option can be used to specify PAPERSIZE= only if the UNIVERSALPRINT and UPRINTMENUSWITCH system options are also specified. For details, see the UNIVERSALPRINT system option in SAS System Options: Reference and the UPRINTMENUSWITCH system option in SAS Companion for Windows.

See Also

Conceptual Information:
- Chapter 2, “Locale for NLS,” on page 5
- “LOCALE= Values for PAPERSIZE and DFLANG, Options” on page 811

System Options:
- “ENCODING System Option: UNIX, Windows, and z/OS” on page 694
- “DATESTYLE= System Option” on page 686
- “DFLANG= System Option: UNIX, Windows, and z/OS” on page 691
- “PAPERSIZE= System Option” in SAS System Options: Reference
LOCALEDATA System Option: UNIX, Windows, and z/OS

Specifies the source database for the locale information.

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS GROUP=
Default: SASLOCALE

Syntax

LOCALEDATA=SASLOCALE | REGISTRY

Syntax Description

SASLOCALE
specifies the locale definition from an internal SAS database.

REGISTRY
specifies the locale definition from the SAS registry.

Details

The SASLOCALE value system option specifies an internal database as the source of the locale’s definition. This database does not allow customization.

The REGISTRY value specifies an external database where locale definitions are customized by using PROC LOCALEDATA.

LOGLANGCHG System Option

Specifies whether the language of the SAS log can be changed.

Valid in: configuration file, SAS invocation
PROC OPTIONS GROUP=
Default: NOLOGLANGCHG

Syntax

LOGLANGCHG | NOLOGLANGCHG
Syntax Description

**LOGLANGCHG**
specifies that the language of the SAS log messages can be changed after start-up.

**NOLOGLANGCHG**
specifies that the language of the SAS log message cannot be changed after start-up.

Details

The following conditions result in these actions:

<table>
<thead>
<tr>
<th>ODSLANGCHG</th>
<th>LOGLANGCHG</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>ON</td>
<td>Language for ODS output and the SAS log can be changed based on LSWLANG= option setting.</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>Language for ODS output and the SAS log cannot be changed.</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>Language for the SAS log can be changed based on LSWLANG= option setting.</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>Language for ODS output can be changed based on LSWLANG= option.</td>
</tr>
</tbody>
</table>

See Also

- “ODSLANGCHG System Option” on page 706
- “LSWLANG System Option” on page 703
- “LOGLANGENG System Option” on page 701

**LOGLANGENG System Option**

Specifies using the English language for SAS log message text when the LOCALE option is set after start-up.

Valid in: configuration file, SAS invocation

Category: Environment Control: Language Control

PROC OPTIONS GROUP=

LANGUAGECONTROL
Default: NOLOGLANGENG

Syntax

LOGLANGENG | NOLOGLANGENG

Syntax Description

LOGLANGENG
SAS log messages are displayed in English using the LSW feature.

NOLOGLANGENG
The LSW feature is not used to enable English for SAS log messages.

Details

The LOGLANGENG option interacts with the LOGLANGCHG, ODSLANGCHG, and LSWLANG options.

- LOGLANGENG has the same functionality as LSWLANG=EN and NOLOGLANGCHG.
- ODSLANGCHG is valid with LOGLANGENG.
- NOLOGLANGENG has no impact on the SAS session.
- During start-up, the configuration file (sasroot\ls) and the LOCALE option determine the language for SAS messages. After start-up, if the LOCALE option is set and the LOGLANGENG option is on, the language for the SAS log output is English.
- If LOGLANGENG is enabled at start-up and the value of LOCALE is changed during the session, the SAS log output remains in English.
- If LOGLANGCHG is on, the SAS log is controlled by the LSWLANG= option.
- If ODSLANGCHG is on, the ODS text is controlled by the LSWLANG= option.
- If LOGLANGENG is not enabled at start-up and LOCALE is changed during the session, the SAS log output is displayed in the language that was set at start-up, unless both LOGLANGCHG and LSWLANG= are specified and LSWLANG has a value other than LOCALE.

Example

This example is a French SAS session with LOGLANGENG set to ON.

If LOGLANGENG is set to ON, then LSWLANG=EN and LOGLANGCHG is ON automatically. The SAS session always uses English for the SAS log, but the language of the ODS output is determined by the current SAS configuration.

See Also

- “LSWLANG System Option” on page 703
- “LOGLANGCHG System Option” on page 700
- “ODSLANGCHG System Option” on page 706
LSWLANG System Option

Specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation.

Valid in: configuration file, SAS invocation
Category: Language Control
PROC OPTIONS GROUP=LANGUAGECONTROL
Default: LOCALE

Syntax

\texttt{LSWLANG=} LOCALE | \textit{language}

Required Arguments

\textbf{LOCALE}

specifying the LOCALE argument preserves the behavior prior to SAS 9.4, where the SAS message text matches the value of the LOCALE= option.

\textbf{language}

Specifies the language for ODS and the SAS log output. The following values can be specified:

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>English</td>
</tr>
<tr>
<td>FR</td>
<td>French</td>
</tr>
<tr>
<td>IT</td>
<td>Italian</td>
</tr>
<tr>
<td>DE</td>
<td>German</td>
</tr>
<tr>
<td>ES</td>
<td>Spanish</td>
</tr>
<tr>
<td>ZH</td>
<td>Chinese S</td>
</tr>
<tr>
<td>ZT</td>
<td>Chinese T</td>
</tr>
<tr>
<td>JA</td>
<td>Japanese</td>
</tr>
<tr>
<td>KO</td>
<td>Korean</td>
</tr>
<tr>
<td>PL</td>
<td>Polish</td>
</tr>
</tbody>
</table>
MAPEBCDIC2ASCII= System Option

Specifies a translation table that SAS uses to transcode from EBCDIC to ASCII and from ASCII to EBCDIC.

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS GROUP= LANGUAGECONTROL

Alias: MAPE2A

Interaction: The MAPEBCDIC2ASCII= system option specifies a translation table to use for the SAS session. The table specified with MAPEBCDIC2ASCII is used by components such as $EBCDIC and $ASCII formats.

Syntax

MAPEBCDIC2ASCII= TRANTAB catalog-entry

Required Argument

catalog-entry
specifies a SAS catalog TRANTAB entry that contains translation tables. If you specify entry-name.type, SAS first searches SASUSER.PROFILE, then SASHELP_LOCALE, and then SASHELP_HOST for the name specified.

Details

MAPEBCDIC2ASCII= supports the requirements of national languages.

**CAUTION**
Do not change a translation table unless you are familiar with its purpose. Translation tables are used internally by SAS to implement NLS. If you are unfamiliar with translation tables, do not change the specifications without proper technical advice.
NLDECSEPARATOR System Option

Specifies whether SAS produces locale sensitive numeric output for the decimal separator or continues to format numbers with US English preferences.

Valid in:
- configuration file
- SAS invocation
- OPTIONS statement
- SAS System Options window

PROC
OPTIONS
GROUP=

Alias: NLD

Default: NONLSDECSEPARATOR

Syntax

NLDECSEPARATOR | NONLDECSEPARATOR

Required Arguments

NLDECSEPARATOR
- Enables locale-sensitive numeric output for the decimal separator.

NONLDECSEPARATOR
- Disables locale-sensitive numeric output for the decimal separator.

Details

Use the BESTDOTX format with the NLDECSEPARATOR option to produce a numeric value with a dot as a decimal separator. If you do not use the BESTDOTX format, the result could contain a comma (1,2). SAS does not interpret this output as a numeric value. It would result in an error if the value is used in a later assignment statement.

Some SAS functions, such as CALL SYMPUT, convert numeric values to a character string using the BEST format by default. When NLDECSEPARATOR is enabled, the BEST format uses the separator character that matches the preferences for the current locale. If the locale uses a dot as a separator character, the resulting string can be used later in a SAS statement that evaluates the value as a number. However, if the separator character is a comma or other character, the result causes a syntax error if the SAS code attempts to use it as a number.

To avoid this type of error, use BESTDOTX to format the value before assigning it within the CALL SYMPUT statement. BESTDOTX always uses a dot as a decimal separator in the results that it produces. For example, in the SAS code here, BESTDOTX is used to format the value of the variable X before it is assigned to the macro variable &MACX. The value of &MACX can then be assigned to the SAS variable Y without causing a syntax error.

```sas
data a;
x=1.2;
   CALL SYMPUT('macx', put(x, bestdotx.));
run;
```
```sas
data b;
  y=&macx
run;
```

See Also

"BESTDOTXw. Format" on page 120

**ODSLANGCHG System Option**

Determines whether the language of the text of the ODS output can be changed.

**Syntax**

```
ODSLANGCHG | NOODSLANGCHG
```

**Details**

The Language Switching feature (LSW) enables you to change the language of SAS messages and ODS templates after start-up. In order to use the LSW for ODS output, you must enable ODSLANGCHG.

During start-up, the configuration file determines the language for SAS messages and ODS templates. If the ODSLANGCHG option is set, the language of ODS output can change to reflect the LSWLANG= setting when the localizations are available.

You can enable ODSLANGCHG but not translate into the language of the locale. For example, if you enable ODSLANGCHG, then start a SAS session in French and set the locale to Greek, NLDATE is displayed in Greek. The output is displayed in French. The output is displayed in French because SAS does not translate into Greek.
Comparisons

If ODSLANGCHG is enabled and LSWLANG=LOCALE is set, the ODS PATH is updated based on the LOCALE setting to include the localized templates and corresponding localized messages are used to generate ODS output.

If ODSLANGCHG is not enabled at start-up, ODS output appears in the language that was specified in the configuration file.

Example

Example 1 is a French server with ODSLANGCHG not enabled (NOODSLANGCHG).

If a French-client application connects to the server, the output appears in French and dates, formatted by using the NL format, appear in French. If an English-client application connects to the French server, and the locale is changed to English on the server, then output messages appears in French, and dates formatted with NL formats appear in English.

Example 2 is a French server with ODSLANGCHG enabled (ODSLANGCHG) and LSWLANG=LOCALE.

If a French-client application connects to the server, the output appears in French and dates formatted by using the NL format, appear in French. If an English-client application connects to the French server, and the locale is changed to English on the server, then output messages appears in English, and dates formatted with NL format appears in English.

Example 3 is a French server with ODSLANGCHG enabled (ODSLANGCHG) and LSWLANG=English.

If a French-client application connects to the server, the output appears in English and dates formatted by using the NL format, appear in French. If an English-client application connects to the French server, and the locale is changed to English on the server, then output messages appear in English, and dates formatted with NL format appears in English.

See Also

- “LOGLANGENG System Option” on page 701
- “LOGLANGCHG System Option” on page 700
- “LSWLANG System Option” on page 703

RSASIOTRANSERR System Option

Displays a transcoding error when invalid data is read from a remote application.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES

Default: RSASIOTRANSERR
Syntax

RSASIOSTRANSERROR | NORSASIOSTRANSERROR

Syntax Description

RSASIOSTRANSERROR
specifies to display a transcoding error when invalid values are read from a remote application.

NORSASIOSTRANSERROR
specifies not to display a transcoding error when invalid values are read from a remote application.

Details

The RSASIOSTRANSERROR system option enables remote users of SASIO, for example, SAS Enterprise Guide and SAS Enterprise Miner, to ignore invalid data values. An invalid data value typically causes a transcoding error when the data is read by a remote application.

SORTSEQ= System Option: UNIX, Windows, and z/OS

Specifies a language-specific collating sequence for the SORT and SQL procedures to use in the current SAS session.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category: Sort: Procedure Options
PROC OPTIONS GROUP=

Syntax

SORTSEQ=<sort-table> | <LINGUISTIC>

Syntax Description

sort-table
specifies a translation table that you created with PROC TRANTAB.

LINGUISTIC
Sorts characters according to the rules of a specified language. The rules and default collating sequence are based on the language specified in the current locale setting. The implementation is provided by the International Components for Unicode (ICU) library. Refer to "Linguistic Sorting of Data Sets and ICU" in Base SAS Procedures Guide for more information.
Example

This example demonstrates the functionality of the SORTSEQ= option with PROC SORT and PROC SQL:

```sas
options sortseq=linguistic;
proc sort data=sashelp.class out=foo1;
  by name;
run;
proc sql;
  create table foo2 as select * from sashelp.class order by name;
quit;
run;
```

SAS provides ICU collation when the linguistic option (SORTSEQ=LINGUISTIC) is specified in the Base SAS procedure, PROC SORT. Linguistic collation can also be specified using the SORTSEQ= option in the SQL procedure and by specifying system option SORTSEQ=LINGUISTIC.

Note: Only PROC SORT and PROC SQL are affected when the SORTSEQ=LINGUISTIC system option is specified.

When the linguistic option is specified, SAS relies on the ICU libraries as the reference implementation of the Unicode Collation Algorithm (UCA) and as a de facto standard. For more information about the UCA algorithm or the ICU library implementation, see Download the ICU 4.8 Release and CLDR 2.0 Release Note.

In SAS Viya, the ICU library version incorporated by SAS and used by PROC SORT is ICU 56. This ICU version uses locale data from version 28 of the Unicode Common Locale Data Repository (CLDR). For in-depth information, see Download ICU 56 and CLDR 28 Release Note.

See Also

- “Collating Sequence” on page 15

System Options:

- "TRANTAB= System Option" on page 711

---

**TIMEZONE= System Option**

Specifies the user local time zone.

**Valid in:**
- SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable (UNIX only), SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable (Linux only)

**Category:**
- Environment Control: Language Control

**PROC OPTIONS GROUP=**
- LANGUAGECONTROL

**Alias:**
- TZ=, except in the restricted options configuration file where TIMEZONE= must be used
This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

Syntax

TIMEZONE= 'time-zone-name' | 'time-zone-ID'

Syntax Description

*time-zone-name*

specifies a three- or four-character time zone name. For example, EST is a time zone name for Eastern Time.

Default

BLANK, indicating that the SAS server time zone and the client time zone are the same

See

For a list of time zone names, see Appendix 3, "Time Zone IDs and Time Zone Names," on page 953.

<->

specifies the time offset from Greenwich, England in the form <->hh:mm:ss>:

- specifies the time zone offset west of Greenwich, England. Time zone offsets east of Greenwich do not require signed notation.

hh specifies an hour by using one or two digits. If you specify an offset, the hour is required.

mm specifies minutes by using one or two digits. Minutes are optional.

ss specifies seconds by using one or two digits. Seconds are optional.

*time-zone-ID*

specifies a region/area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by time zone name and daylight saving time rules.

Note

Time zone IDs are compatible with Java time zone names.

See

For a list of time zone IDs, see Appendix 3, "Time Zone IDs and Time Zone Names," on page 953.

Details

You set the TIMEZONE= option to a time zone ID or a time zone name in order for SAS to use a particular time zone. The time zone setting affects the following SAS components:

- times that are recorded by events and logs
- time of data set creation or modification
- DATE( ) function
- DATETIME( ) function
- TIME( ) function
- TODAY( ) function
time zone functions TZONEOFF( ), TZONEID( ), TZONENAME( ), TZONES2U( ), and TZONEU2S( ),

time zone formats B8601DXw., E8601DXw., B8601LXw., E8601LXw., B8601TXw., E8601TXw., NLDATMZw., NLDATMTZw., and NLDATMWZw.

You set a time zone by specifying a time zone ID or a time zone name. A time zone ID is a region and an area separated by a forward slash (/). For example, America/New_York and Asia/Osaka are time zone IDs.

A time zone name is a three- or four-character name for a time zone. For example, EST is Eastern Time and JST is Japan Time. SAS determines the time by using time zone rules, including daylight saving time rules, before using a time value.

Some time zones names are valid for different locales. For example, CST is Central Standard Time, Cuba Daylight Time, and China Standard Time. SAS uses the value of the LOCALE= system option to determine the region and area to use. If TIMEZONE="CST" and LOCALE="zh_CN", SAS uses the Asia/Beijing time zone. If the time zone name does not exist for the locale, SAS searches all time zones and sets the time zone to the first match that it finds.

When this option is restricted and the value of TIMEZONE= is the default value of BLANK, time zone behavior does not use time zone information.

See Also

“Specifying Time Zones in SAS” on page 51

TRANTAB= System Option

Specifies the translation tables that are used by various parts of SAS.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment Control: Language Control

PROC OPTIONS GROUP=

Defaults: SAS 9.4: The shipped default is (lat1wlt1,wlt1lat1,lat1_ucs,lat1_lcs,lat1_ccl,,)
SAS Viya: The shipped default is (lat1wlt1,wlt1lat1,lat1_ucs,lat1_lcs,lat1_ccl,,)

Interaction: The TRANTAB= system option specifies a translation table to use for the SAS session, including file transfers. The TRANTAB statement specifies a customized translation table (for example, to map an EBCDIC character to an ASCII character) to apply to the character set in the SAS file that is being exported or transferred.

Syntax

TRANTAB=(catalog-entries)

Syntax Description

catalog-entries

specifies SAS catalog entries that contain translation tables. If you specify entry-name.type, SAS searches SASUSER.PROFILE first and then SASUSER.HOST.
Details

TRANTAB= was introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on the features of TRANTAB=. In SAS 9, translations tables are set by the LOCALE= system option. They are used for transcoding external files. For SAS files, there is a direct transcoding between the session encodings.

SAS 9.2 supports the TRANTAB procedure for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases.

You can list the translation tables with PROC OPTIONS. The following example is on the z/OS environment with a LOCALE option set to en_US.

```
TRANTAB=(eol1wlt1,wlt1eol1,elat_ucs,elat_lcs,elat_ccl,,,elat_scc)
```

Translation tables are specified in a list that is enclosed in parentheses and has ten positions. The position in which a table appears in the list determines the type of translation table that is specified. Individual entries in the list are separated by commas. See the list of positions and types that follows:

<table>
<thead>
<tr>
<th>Position</th>
<th>Type of Translation Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>local-to-transport-format</td>
</tr>
<tr>
<td>2nd</td>
<td>transport-to-local-format</td>
</tr>
<tr>
<td>3rd</td>
<td>lowercase-to-uppercase</td>
</tr>
<tr>
<td>4th</td>
<td>uppercase-to-lowercase</td>
</tr>
<tr>
<td>5th</td>
<td>character classification</td>
</tr>
<tr>
<td>6th</td>
<td>scanner translation</td>
</tr>
<tr>
<td>7th</td>
<td>delta characters</td>
</tr>
<tr>
<td>8th</td>
<td>scanner character classification</td>
</tr>
<tr>
<td>9th</td>
<td>not used</td>
</tr>
<tr>
<td>10th</td>
<td>DBCS user table</td>
</tr>
</tbody>
</table>

**CAUTION**

*Do not change a translation table unless you are familiar with its purpose.*

Translation tables are used internally by the SAS supervisor to implement NLS. If you are unfamiliar with the purpose of translation tables, do not change the specifications without proper technical advice.

To change one table, specify null entries for the other tables. For example, to change the lowercase-to-uppercase table, which is third in the list, specify uppercase as follows:
options trantab = ( , , new-uppercase-table);

The other tables remain unchanged. The output from the OPTIONS procedure reflects the last specification for the TRANTAB= option and not the composite specification. Here is an example:

options trantab = ( , , new-uppercase-table);
options trantab = ( , , , new-lowercase-table);

PROC OPTIONS shows that the value for TRANTAB= is
( , , new-lowercase-table), but both the new-uppercase and new-lowercase tables are in effect.

See Also
Chapter 20, “TRANTAB Procedure,” on page 783
Names must begin with a letter of the Latin alphabet (A–Z, a–z) or an underscore. Subsequent characters can be letters of the Latin alphabet, numerals, or underscores.

Names cannot contain blanks or special characters except for the underscore.

Names can contain mixed-case letters. SAS internally converts the member name to uppercase. Therefore, you cannot use the same member name with a different combination of uppercase and lowercase letters to represent different variables. For example, customer, Customer, and CUSTOMER all represent the same member name. How the name is saved on disk is determined by the operating environment.

Alias COMPAT

EXTEND

specifies that a SAS data set name, a SAS data view name, or an item store name must follow these rules:

- Names can include national characters.
- The name can include special characters, except for the / \ * ? " < > |: -. characters.

Note: The SPD Engine does not allow ‘.’ (the period) anywhere in the member name.

- The name must contain at least one character (letters, numbers, valid special characters, and national characters).
- The length of the name can be up to 32 bytes.
- Null bytes are not allowed.
- Names cannot begin with a blank or a ‘.’ (the period).

Note: The SPD Engine does not allow ‘$’ as the first character of the member name.

- Leading and trailing blanks are deleted when the member is created.
- Names can contain mixed-case letters. SAS internally converts the member name to uppercase. Therefore, you cannot use the same member name with a different combination of uppercase and lowercase letters to represent different variables. For example, customer, Customer, and CUSTOMER all represent the same member name. How the name appears is determined by the operating environment.

Requirement When VALIDMEMNAME=EXTEND, SAS data set names, SAS data view names, and item store names must be written as a SAS name literal if the name includes blank spaces, special characters, or national characters. If you use either the percent sign (%) or the ampersand (&), then you must use single quotation marks in the name literal in order to avoid interaction with the SAS Macro Facility. For more information, see “SAS Words” in SAS Programmer’s Guide: Essentials.

Note: The SPD Engine does not allow ‘.’ (the period) anywhere in the member name.
For Windows and UNIX operating environments, all Base SAS windows support the extended rules when
VALIDMEMNAME=EXTEND is set.

For Windows and UNIX operating environments, when you reference a SAS file directly by its physical name, the final
embedded period is an extension delimiter. If a physical file reference includes a SAS member name that contains a period,
you must add the file extension. For example, if you reference the data set name my.member as a physical file, you would add
the file extension sas7bdat to the reference, as shown in this SET statement: set './saslib/my.member.sas7bdat'.

The windowing environment for Base SAS supports the extended rules in the Editor, Log, and Output windows when
VALIDMEMNAME=EXTEND is set. Other SAS windows, such as the VIEWTABLE window, do not support the extended rules.

When you reference a SAS file directly by its physical name, the final embedded period is considered to be an extension
delimiter only if what follows the period is a valid SAS extension. Otherwise, the period is considered to be part of the member
name. For example, in the name my.member, member is considered part of the member name and not a file extension. In
the name 'my.member.sas7bdat', the member name is 'my.member' and the file extension is sas7bdat.

The name is displayed in uppercase letters.


Examples

data "August Purchases"
;

data 'Años de empleo'
;
support the extended characters. INDEX and AUDIT types that exist only with the associated DATA member support extended characters.

See Also


System Options:

- “VALIDVARNAME= System Option” on page 716

---

**VALIDVARNAME= System Option**

Specifies the rules for valid SAS variable names that can be created and processed during a SAS session.

Valid in: SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SAV9_OPTIONS environment variable (UNIX only), SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SAV9_OPTIONS environment variable (Linux only)

Category: Files: SAS Files

PROC OPTIONS GROUP=SASFILES

Defaults: The SAS shipped default is V7.
The SAS Viya shipped default is ANY.

Interaction: SAS Studio sets VALIDVARNAME= to the Preferences setting before each code submission. For more information, see “System Options in SAS Studio” in SAS System Options: Reference.

Note: This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

Syntax

```
VALIDVARNAME=V7 | UPCASE | ANY
```

Syntax Description

**V7**

specifies that variable names must follow these rules:

- The name can be up to 32 characters.
- The first character must begin with a letter of the Latin alphabet (A - Z, a - z) or the underscore. Subsequent characters can be letters of the Latin alphabet, numerals, or underscores.
- Trailing blanks are ignored. The variable name alignment is left-justified.
- A variable name cannot contain blanks or special characters except for the underscore.
A variable name can contain mixed-case letters. SAS stores and writes the variable name in the same case that is used in the first reference to the variable. However, when SAS processes a variable name, SAS internally converts it to uppercase. Therefore, you cannot use the same variable name with a different combination of uppercase and lowercase letters to represent different variables. For example, \texttt{cat}, \texttt{Cat}, and \texttt{CAT} all represent the same variable.

Do not assign variables the names of special SAS automatic variables (such as \_N\_ and \_ERROR\_) or variable list names (such as \_NUMERIC\_, \_CHARACTER\_, and \_ALL\_) to variables.

**Examples**

```
season='summer';
percent_of_profit=percent;
```

**UPCASE**

specifies that the variable name follows the same rules as V7, except that the variable name is uppercase, as in earlier versions of SAS.

**ANY**

specifies that SAS variable names must follow these rules:

- The name can begin with or contain any characters, including blanks, national characters, special characters, and multi-byte characters.
- The name can be up to 32 bytes in length.
- The name cannot contain any null bytes.
- Leading blanks are preserved, but trailing blanks are ignored.
- The name must contain at least one character. A name with all blanks is not permitted.
- The name can contain mixed-case letters. SAS stores and writes the variable name in the same case that is used in the first reference to the variable. However, when SAS processes a variable name, SAS internally converts it to uppercase. Therefore, you cannot use the same variable name with a different combination of uppercase and lowercase letters to represent different variables. For example, \texttt{cat}, \texttt{Cat}, and \texttt{CAT} all represent the same variable.

**Requirement**

If you use any characters other than the ones that are valid when the \texttt{VALIDVARNAME} system option is set to V7 (letters of the Latin alphabet, numerals, or underscores), then you must express the variable name as a name literal and you must set \texttt{VALIDVARNAME=ANY}. If the name includes either the percent sign (\%) or the ampersand (\&), then you must use single quotation marks in the name literal in order to avoid interaction with the SAS Macro Facility. See “SAS Names” in \textit{SAS Programmer’s Guide: Essentials} and “Avoiding Errors When Using Name Literals” in \textit{SAS Programmer’s Guide: Essentials}.

**See**


**Examples**

```
'\% of profit'n=percent;

'items@warehouse'n=itemnum;
```
CAUTION Throughout SAS, using the name literal syntax with SAS variable names that exceed the 32-byte limit or have excessive embedded quotation marks might cause unexpected results. The intent of the VALIDVARNAME=ANY system option is to enable compatibility with other DBMS variable (column) naming conventions, such as allowing embedded blanks and national characters.

See Also


System Options:

- “VALIDMEMNAME= System Option” on page 713

**URLENCODING= System Option**

Specifies whether the argument to the URLENCODE function and to the URLDECODE function is interpreted using the SAS session encoding or UTF-8 encoding.

Valid in: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment Control: Language Control

PROC OPTIONS GROUP=

See: “URLENCODING= System Option” in SAS System Options: Reference

**Syntax**

**URLENCODING=** SESSION | UTF8
Options for Commands, Statements, and Procedures for NLS
Dictionary of Command, Statement, and Procedure Option for NLS

Commands, Statements, and Procedures for NLS by Category

The data set control and data access categories of options for selected SAS statements are affected by NLS. The following table provides brief descriptions of the statement options. For more detailed descriptions, see the dictionary entry for each statement option:

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Access</td>
<td>CVPBYTES=, CVPENGINE=, CVPFORMATWIDTH=, CVMULTIPLIER=, CVPVARCHAR=, CVPEXCLUDE=,</td>
<td>Specifies attributes for character variables that are needed to transcode a SAS file.</td>
</tr>
<tr>
<td></td>
<td>CVPINCLUDE=</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(p. 731)</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>ENCODING= Option Statement (p. 742)</td>
<td>Overrides and transcodes the encoding for input or output processing of external files.</td>
</tr>
<tr>
<td></td>
<td>INENCODING= StatementOUTENCODING= Options Statement (p. 745)</td>
<td>Overrides and changes the encoding when reading or writing SAS data sets in the SAS library.</td>
</tr>
<tr>
<td></td>
<td>ODSCHARSET= Option Statement (p. 747)</td>
<td>Specifies the character set to be generated in the META declaration for the output.</td>
</tr>
<tr>
<td></td>
<td>ODSTRANTAB= Option Statement (p. 748)</td>
<td>Specifies the translation table to use when transcoding an XML document for an output file.</td>
</tr>
<tr>
<td></td>
<td>RENCODING= Option Statement (p. 749)</td>
<td>Specifies the ASCII-based or EBCDIC-based encoding to use for transcoding data for a SAS/SHARE server session that is using an EBCDICANY or ASCIIANY session encoding.</td>
</tr>
<tr>
<td></td>
<td>XMLENCODING= Option Statement (p. 754)</td>
<td>Overrides the encoding of an XML document to import or export an external document.</td>
</tr>
<tr>
<td>Information</td>
<td>TRANSCODE= Option Statement (p. 751)</td>
<td>Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and length with one or more variables) that indicates whether character variables are to be transcoded.</td>
</tr>
<tr>
<td>ODS: Third-Party Formatted</td>
<td>CHARSET= Statement (p. 722)</td>
<td>Specifies the character set to be generated in the META declaration for the output.</td>
</tr>
<tr>
<td></td>
<td>TRANTAB= Option Statement (p. 753)</td>
<td>Specifies the translation table to use when you are transcoding character data.</td>
</tr>
</tbody>
</table>

**Dictionary**

**CHARSET= Statement**

Specifies the character set to be generated in the META declaration for the output.

**Valid in:** LIBNAME statement for the ODS MARKUP and ODS HTML statements  
**Category:** ODS: Third-Party Formatted

**Syntax**

```
CHARSET=character-set ;
```
Required Argument

**character-set**

Specifies the character set to use in the META tag for HTML output.

An example of an encoding is UTF-8. Official character sets for use on the internet are registered by IANA (Internet Assigned Numbers Authority). IANA is the central registry for various internet protocol parameters, such as port, protocol and enterprise numbers, and options, codes and types. For a complete list of character-set values, see [www.unicode.org/reports/tr22/index.html](http://www.unicode.org/reports/tr22/index.html) and [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets).

A character set is like an encoding-value in this context. However, character set is the term that is used to identify an encoding that is suitable for use on the internet.

Example: Generated Output in a META Declaration for an ODS MARKUP Statement

```html
<META http-equiv="Content-Type" content="text/html; charset=utf-8">
```

See Also

- **Conceptual Information:**
  - Chapter 3, “Encoding for NLS,” on page 9

- **Statements:**

---

**Collating Sequence Option Statement**

Specifies the collating sequence for PROC SORT.

**Valid in:**

- PROC SORT statement

**Note:**

The PROC SORT statement sorts observations in a SAS data set by one or more characters or numeric variables.

**Syntax**

```
PROC SORT collating-sequence-option <other options> ;
```

**Options**

Options can include one `collating-sequence-option` and multiple `other options`. The order of the two types of options does not matter and both types are not necessary in the same PROC SORT step. Only the explanations for the PROC SORT collating-sequence-options follow.
Operating Environment Information: For information about behavior specific to your operating environment for the DANISH, FINNISH, NORWEGIAN, or SWEDISH collating-sequence-option, see the SAS documentation for your operating environment.

**ASCII**
sorts character variables using the ASCII collating sequence. You need this option only when you want to achieve an ASCII ordering on a system where EBCDIC is the native collating sequence.

**DANISH NORWEGIAN**
sorts characters according to the Danish and Norwegian

The Danish and Norwegian collating sequence is shown in Figure 17.6 on page 725.

**EBCDIC**
sorts character variables using the EBCDIC collating sequence. You need this option only when you want to achieve an EBCDIC ordering on a system where ASCII is the native collating sequence.

**POLISH**
sorts characters according to the Polish convention.

**FINNISH SWEDISH**
sorts characters according to the Finnish and Swedish convention. The Finnish and Swedish collating sequence is shown in Figure 17.6 on page 725.

**NORWEGIAN**
See DANISH

**SWEDISH**
See FINNISH

**SORTSEQ=collating-sequence**
specifies the collating sequence. The collating-sequence can be a collating-sequence-option, a translation table, an encoding, or the keyword LINGUISTIC. Only one collating sequence can be specified. For more information, see "Collating Sequence" on page 15.

Here are descriptions of the collating sequences:

**collating—sequence—option | translation_table**
specifies either a translation table, which can be one that SAS provides or any user-defined translation table, or one of the PROC SORT statement Collating-Sequence-Options. For an example of using PROC TRANTAB and PROC SORT with SORTSEQ=, see "Example 6: Using Different Translation Tables for Sorting" on page 803.

The available translation tables are

- ASCII
- DANISH
- EBCDIC
- FINNISH
- ITALIAN
- NORWEGIAN
- POLISH
- REVERSE
The following figure shows how the alphanumeric characters in each language sorts:

Figure 17.1 Alphanumeric Characters Sorted for Each Language

<table>
<thead>
<tr>
<th>Language</th>
<th>Sorted Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Danish</td>
<td>0123456789A...</td>
</tr>
<tr>
<td>Finnish</td>
<td>0123456789A...</td>
</tr>
<tr>
<td>Italian</td>
<td>0123456789A...</td>
</tr>
<tr>
<td>Norwegian</td>
<td>0123456789A...</td>
</tr>
<tr>
<td>Spanish</td>
<td>0123456789A...</td>
</tr>
<tr>
<td>Swedish</td>
<td>0123456789A...</td>
</tr>
</tbody>
</table>

Restriction: You can specify only one collating-sequence-option in a PROC SORT step.

Tip: The SORTSEQ= collating sequence options are specified without parenthesis and have no arguments that are associated with them. An example of how to specify a collating sequence follows:

```
proc sort data=mydata SORTSEQ=ASCII;
```

`encoding-value` specifies an encoding value. The result is the same as a binary collation of the character data represented in the specified encoding. See the supported encoding values in "SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" on page 829.

Restriction: PROC SORT is the only procedure or part of the SAS system that recognizes an encoding specified for the SORTSEQ= option.

Tip: When the encoding value contains a character other than an alphanumeric character or underscore, the value needs to be enclosed in quotation marks.

See: The list of the encodings that can be specified in "SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" on page 829.

LINGUISTIC>(collating—rules )> specifies linguistic collation, which sorts characters according to rules of the specified language. The rules and default collating sequence options are based on the language specified in the current locale setting. The implementation is provided by the International Components for Unicode (ICU) library and produces results that are largely compatible with the Unicode Collation Algorithms (UCA).

Alias: UCA

Restriction: The SORTSEQ=LINGUISTIC option is available only on PROC SORT and PROC SQL. Collating rules are supported for PROC SORT, not for the system option or SORTSEQ= on PROC SQL.
Note  Linguistic collation can be specified using the SORTSEQ= option in the SQL Procedure and by specifying system option SORTSEQ=LINGUISTIC. For more information, see “SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 708 and “SORT Procedure” in Base SAS Procedures Guide.

Tips  LINGUISTIC sorting requires more memory with the z/OS mainframe. You might need to set your REGION to 50M or higher. This action must be done in JCL, if you are running in batch mode, or in the VERIFY screen if you are running interactively. This action allows the ICU libraries to load properly and does not affect the memory that is used for sorting.

The collating-rules must be enclosed in parentheses. More than one collating rule can be specified.

When BY processing is performed on data sets that are sorted with linguistic collation, the NOBYSORTED system option might need to be specified in order for the data set to be treated properly. BY processing is performed differently than collating sequence processing.

See  The “ICU License” in SAS Companion for z/OS

The “Collating Sequence” on page 15 for detailed information about linguistic collation.

“SORTSEQ=sort-table | LINGUISTIC” in SAS SQL Procedure User’s Guide for information about linguistic sorting in PROC SORT.


Here are the collation-rules that can be specified for the LINGUISTIC option. These rules modify the linguistic collating sequence:

**ALTERNATE_HANDLING=SHIFTED**
controls the handling of variable characters like spaces, punctuation, and symbols. When this option is not specified (using the default value NON_IGNORABLE), differences among these variable characters are of the same importance as differences among letters. If the ALTERNATE_HANDLING option is specified, these variable characters are of minor importance.

Default  NON_IGNORABLE

Tip  The SHIFTED value is often used in combination with STRENGTH= set to Quaternary. In such a case, whitespace characters, punctuation, and symbols are considered when comparing strings, but only if all other aspects of the strings (base letters, accents, and case) are identical.

**CASE_FIRST=**
specify order of uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL levels. The following table provides the values and information for the CASE_FIRST argument:
### Collating Sequence Option Statement

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UPPER</strong></td>
<td>Sorts uppercase letters first, then the lowercase letters.</td>
</tr>
<tr>
<td><strong>LOWER</strong></td>
<td>Sorts lowercase letters first, then the uppercase letters.</td>
</tr>
</tbody>
</table>

**COLLATION=**

The following table lists the available COLLATION= values: If you do not select a collation value, then the user’s locale-default collation is selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BIG5HAN</strong></td>
<td>specifies pinyin ordering for Latin and specifies big5 charset ordering for</td>
</tr>
<tr>
<td><strong>DIRECT</strong></td>
<td>specifies a Hindi variant.</td>
</tr>
<tr>
<td><strong>GB2312HAN</strong></td>
<td>specifies pinyin ordering for Latin and specifies gb2312han charset ordering</td>
</tr>
<tr>
<td><strong>PHONEBOOK</strong></td>
<td>specifies a telephone-book style for ordering of characters. Select PHONEBOOK</td>
</tr>
<tr>
<td></td>
<td>only with the German language.</td>
</tr>
<tr>
<td><strong>PINYIN</strong></td>
<td>specifies an ordering for Chinese, Japanese, and Korean characters based</td>
</tr>
<tr>
<td></td>
<td>on character-by-character transliteration into pinyin. This ordering is</td>
</tr>
<tr>
<td></td>
<td>typically used with simplified Chinese.</td>
</tr>
<tr>
<td><strong>POSIX</strong></td>
<td>is the Portable Operating System Interface. This option specifies a &quot;C&quot;</td>
</tr>
<tr>
<td></td>
<td>locale ordering of characters.</td>
</tr>
<tr>
<td><strong>STROKE</strong></td>
<td>specifies a nonalphabetic writing style ordering of characters. Select STROKE</td>
</tr>
<tr>
<td></td>
<td>with Chinese, Japanese, Korean, or Vietnamese languages. This ordering is</td>
</tr>
<tr>
<td></td>
<td>typically used with Traditional Chinese.</td>
</tr>
<tr>
<td><strong>TRADITIONAL</strong></td>
<td>specifies a traditional style for ordering of characters. For example, select</td>
</tr>
<tr>
<td></td>
<td>TRADITIONAL with the Spanish language.</td>
</tr>
</tbody>
</table>
**LOCALE=locale_name**

specifies the locale name in the form of a POSIX name. For example, ja_JP. See the "LOCALE= Values for PAPERSIZE and DFLANG, Options" on page 811 for a list of locale and POSIX values supported by PROC SORT.

*Restriction*  The following locales are not supported by PROC SORT:
- Afrikaans_SouthAfrica, af_ZA
- Cornish_GreatBritain, kw_GB

**NUMERIC_COLLATION=**

orders integer values within the text by the numeric value instead of characters used to represent the numbers.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Order numbers by the numeric value. For example, &quot;8 Main St.&quot; would sort before &quot;45 Main St.&quot;.</td>
</tr>
<tr>
<td>OFF</td>
<td>Order numbers by the character value. For example, &quot;45 Main St.&quot; would sort before &quot;8 Main St.&quot;.</td>
</tr>
</tbody>
</table>

*Default*  OFF

**STRENGTH=**

The value of strength is related to the collation level. There are five collation-level values. The following table provides information about the five levels. The default value for strength is related to the locale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Type of Collation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY or 1</td>
<td>PRIMARY specifies differences between base characters (for example, &quot;a&quot; &lt; &quot;b&quot;).</td>
<td>It is the strongest difference. For example, dictionaries are divided into different sections by base character.</td>
</tr>
<tr>
<td>SECONDARY or 2</td>
<td>Accents in the characters are considered secondary differences (for example, &quot;as&quot; &lt; &quot;às&quot; &lt; &quot;at&quot;).</td>
<td>A secondary difference is ignored when there is a primary difference anywhere in the strings. Other differences between letters can also be considered secondary differences, depending on the language.</td>
</tr>
<tr>
<td>Value</td>
<td>Type of Collation</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TERTIARY or 3</td>
<td>Upper and lowercase differences in characters are distinguished at the tertiary level (for example, &quot;ao&quot; &lt; &quot;Ao&quot; &lt; &quot;aò&quot;).</td>
<td>A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings. Another example is the difference between large and small Kana.</td>
</tr>
<tr>
<td>QUATERNARY or 4</td>
<td>When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, &quot;ab&quot; &lt; &quot;a-b&quot; &lt; &quot;aB&quot;).</td>
<td>The quaternary level should be used if ignoring punctuation is required or when processing Japanese text. This difference is ignored when there is a primary, secondary, or tertiary difference.</td>
</tr>
<tr>
<td>IDENTICAL or 5</td>
<td>When all other levels are equal, the identical level is used as a tiebreaker. The Unicode code point values of the Normalization Form D (NFD) form of each string are compared at this level, just in case there is no difference at levels 1-4.</td>
<td>This level should be used sparingly, as only code point values differences between two strings is an extremely rare occurrence. For example, only Hebrew cantillation marks are distinguished at this level.</td>
</tr>
</tbody>
</table>

**CAUTION**

If you use a host sort utility to sort your data, then specifying a translation table based collating sequence with the SORTSEQ= option might corrupt the character BY variables. For more information, see the PROC SORT documentation for your operating environment.

**Details**

The collating sequence option in the PROC SORT statement sorts observations in a SAS data set by one or more characters or numeric variables.

**Table 17.1  Options**

<table>
<thead>
<tr>
<th>Task</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the collating sequence</td>
<td></td>
</tr>
<tr>
<td>Specify ASCII</td>
<td>ASCII on page 724</td>
</tr>
<tr>
<td>Task</td>
<td>Option</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Specify EBCDIC</td>
<td>EBCDIC on page 724</td>
</tr>
<tr>
<td>Specify Danish</td>
<td>DANISH on page 724</td>
</tr>
<tr>
<td>Specify Finnish</td>
<td>FINNISH on page 724</td>
</tr>
<tr>
<td>Specify Norwegian</td>
<td>NORWEGIAN on page 724</td>
</tr>
<tr>
<td>Specify Polish</td>
<td>POLISH on page 724</td>
</tr>
<tr>
<td>Specify Swedish</td>
<td>SWEDISH on page 724</td>
</tr>
<tr>
<td>Specify a customized sequence</td>
<td>NATIONAL on page 724</td>
</tr>
<tr>
<td>Specify any of the collating</td>
<td>SORTSEQ= on page 724</td>
</tr>
<tr>
<td>sequences listed above (ASCII,</td>
<td></td>
</tr>
<tr>
<td>EBCDIC, DANISH, DANISH, FINNISH,</td>
<td></td>
</tr>
<tr>
<td>ITALIAN, NORWEGIAN, POLISH,</td>
<td></td>
</tr>
<tr>
<td>SPANISH, SWEDISH, or NATIONAL),</td>
<td></td>
</tr>
<tr>
<td>the name of any other system</td>
<td></td>
</tr>
<tr>
<td>provided translation table</td>
<td></td>
</tr>
<tr>
<td>(POLISH, SPANISH), and the name</td>
<td></td>
</tr>
<tr>
<td>of a user-created translation</td>
<td></td>
</tr>
<tr>
<td>table. You can specify an</td>
<td></td>
</tr>
<tr>
<td>encoding. You can also specify</td>
<td></td>
</tr>
<tr>
<td>either the keyword LINGUISTIC or</td>
<td></td>
</tr>
<tr>
<td>UCA to achieve a locale-appropriate</td>
<td></td>
</tr>
<tr>
<td>collating sequence.</td>
<td></td>
</tr>
</tbody>
</table>

See Also

- “Collating Sequence” on page 15
- “SORT Procedure” in *Base SAS Procedures Guide*

System Options:

- “SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 708
- “TRANTAB= System Option” on page 711

**CORRECTENCODING= Option Statement**

Explicitly changes the encoding attribute of a SAS file to match the encoding of the data in the SAS file.

Valid in: MODIFY statement of the DATASETS procedure

**Syntax**

```sas
MODIFY SAS file <CORRECTENCODING=encoding-value> ;
```
Optional Argument

`<CORRECTENCODING=encoding-value>`

enables you to change the encoding indicator, which is recorded in the file's descriptor information, in order to match the actual encoding of the file's data. You cannot use this option in parenthesis after the name of each SAS file; you must specify CORRECTENCODING= after the forward slash. For example:

```
modify mydata / correctencoding=latin2;
```

For a list of valid encoding values for transcoding, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

Restriction CORRECTENCODING= can be used only when the SAS file uses the default base engine, which is V9 in SAS 9.

Example: Using the CORRECTENCODING= Option to Resolve a SAS Session Encoding and a SAS File Encoding

A file's encoding indicator can be different from the data's encoding. For example, a SAS file that was created before SAS 9 has no encoding indicator stored on the file. If such a SAS file that has no recorded encoding is opened in a SAS 9 session, SAS assigns the encoding of the current session. For example, if the encoding of the data is Danish EBCDIC, but the encoding for the current session is Western Wlatin1, then the actual encoding of the file's data and the encoding indicator that is stored in the file's descriptor information do not match. When this action occurs, the data does not transcode correctly and could result in unreadable output. The following MODIFY statement would resolve the problem by explicitly assigning an EBCDIC encoding:

```
proc datasets library=myfiles;
    modify olddata / correctencoding=ebcdic1142;
quit;
```

Note: CEDA creates a read-only copy. You need to copy the data with PROC COPY or a DATA step to transcode the data permanently.

CVPBYTES=, CVPENGINE=, CVPFORMATWIDTH=, CVPMULTIPLIER=, CVPVARCHAR=, CVPEXCLUDE=, CVPINCLUDE= Statement

Specifies attributes for character variables that are needed to transcode a SAS file.

Valid in: LIBNAME statement
Category: Data Access
PROC OPTIONS GROUP= LIBNAME statement under Windows, UNIX, and Z/OS in the documentation for your operating environment.
Restriction: The CVPEXCLUDE= and CVPINCLUDE= options are available only in SAS Viya 3.5.
See: LIBNAME, SAS/ACCESS
Syntax

`LIBNAME libref 'SAS data-library' <CVPBYTE=bytes> <CVPENGINE=engine> <CVPCHAR=variable expression> <CVPFORMATWIDTH=YES|NO> <CVPINCLUDE=variable expression> <CVPMULTIPLIER=multiplier> <CVPVARCHAR=YES | NO>;`

Optional Arguments

**CVPBYTE=bytes**

specifies the number of bytes by which to expand character variable lengths when processing a SAS data file that requires transcoding. The CVP engine expands the lengths so that character data truncation does not occur. The lengths for character variables are increased by adding the specified value to the current length. You can specify a value from 0 to 32,766.

For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPBYTE= option:

```
libname expand 'SAS data-library' cvpbytes=5;
```

Character variable lengths are increased by adding 5 bytes. A character variable with a length of 10 is increased to 15, and a character variable with a length of 100 is increased to 105.

**Default**

If you specify CVPBYTE=, SAS automatically uses the CVP engine to expand the character variable lengths according to your specification. If you explicitly assign the CVP engine but do not specify either CVPBYTE= or CVMULTIPLIER=, then SAS uses CVMULTIPLIER=1.5 to increase the lengths of the character variables.

**Restrictions**

The CVP engine supports SAS data files, no SAS views, catalogs, item stores, and so on.

The CVP engine is available for input (read) processing only.

For library concatenation with mixed engines that include the CVP engine, only SAS data files are processed. For example, if you execute the COPY procedure, only SAS data files are copied.

**Requirement**

The number of bytes that you specify must be large enough to accommodate any expansion. Otherwise, truncation occurs, which results in an error message in the SAS log.

**Interaction**

You cannot specify both the CVPBYTE= option and the CVMULTIPLIER= option. Specify only one of these options.

**See**

"Avoiding Character Data Truncation By Using the CVP Engine" on page 41

**CVPENGINE=engine**

specifies the engine to use to process a SAS data file that requires transcoding. The CVP engine expands the character variable lengths to transcoding so that character data truncation does not occur. Then the specified engine processes the actual file.
**CVPINCLUDE=** *variable expression*

specifies the variables that are included during data processing. Only the specified variables can be processed.

CVPINCLUDE= and CVPEXCLUDE= cannot be used together. If both options are omitted, then all variables are processed.

**CVPMULTIPLIER=** *multiplier*

specifies a multiplier value that expands character variable lengths when you are processing a SAS data file that requires transcoding. The CVP engine expands the lengths so that character data truncation does not occur. The lengths for character variables are increased by multiplying the current length by the specified value. You can specify a multiplier value from 1 to 5 or you can specify 0 and then the CVP engine determines the multiplier automatically.

For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPMULTIPLIER= option:

```sas
libname expand 'SAS data-library' cvpmultiplier=2.5;
```
Character variable lengths are increased by multiplying the lengths by 2.5. A character variable with a length of 10 is increased to 25, and a character variable with a length of 100 is increased to 250.

<table>
<thead>
<tr>
<th>Alias</th>
<th>CVPMULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>If you specify the CVPMULTIPLIER= option, SAS automatically uses the CVP engine to expand the character variable lengths according to your specification. If you explicitly specify the CVP engine but do not specify either the CVPMULTIPLIER= option or the CVPBYTES= option, then SAS uses CVPMULTIPLIER=AUTO(0) to increase the lengths. AUTO(0) sets the value of the CVP engine based on the encoding of the SAS session and input data set.</td>
</tr>
<tr>
<td>Restrictions</td>
<td>The CVP engine supports SAS data files, no SAS views, catalogs, item stores, and so on. The CVP engine is available for input (read) processing only. For library concatenation with mixed engines that include the CVP engine, only SAS data files are processed. For example, if you execute the COPY procedure, only SAS data files are copied.</td>
</tr>
<tr>
<td>Requirement</td>
<td>The number of bytes that you specify must be large enough to accommodate any expansion. Otherwise, truncation occurs, which results in an error in the SAS log.</td>
</tr>
<tr>
<td>Interaction</td>
<td>You cannot specify both the CVPMULTIPLIER= option and the CVPBYTES= option. Specify only one of these options.</td>
</tr>
<tr>
<td>See</td>
<td>“Avoiding Character Data Truncation By Using the CVP Engine” on page 41</td>
</tr>
</tbody>
</table>

**CVPVARCHAR=YES | NO**

specifies whether to convert fixed-width character variables to variable-width characters during input file processing. The byte length of the new-width character variable is the maximum number of bytes per character from the SAS session encoding multiplied by the specified fixed-width character length.

| Default | No |
| Interaction | If you specify CVPVARCHAR=YES, the CVPMULTIPLIER= and CVPBYTES= options are ignored. |
| Notes | Trailing blanks are removed from string data that is under CHAR columns. Fixed-width character variables with a format of TRANSCODE=NO are excluded during conversion. |

**Details**

The character variable padding (CVP) engine converts variables defined with the CHAR to a VARCHAR data type when CVPVARCHAR=YES. Because CVP is available only for input processing, the VARCHAR data type is not automatically saved.
Without saving the data, the conversion to VARCHAR is lost when a SAS session ends. To save the changes, use the SET statement or PROC COPY and an engine that supports VARCHAR. If the data is saved using an engine that does not support VARCHAR, such as the BASE engine, the character columns in the new data set revert to the CHAR data type. The length of the CHAR variables in the new data set is the number of bytes that is needed to store the VARCHAR.

For more information about VARCHAR, see “VARCHAR Data Type in String Functions” on page 371.

If the data is read by a procedure that does not support VARCHAR, CVPVARCHAR=YES is ignored. The CVP engine uses CVPBYTES, CVPMULTIPLIER, or the default multiplier to expand the length of the character columns in the data.

By default, the CVP engine uses a multiplier of 1.5 times the variable length. This expansion is usually sufficient to support the data for Western European languages because the text for those languages usually has a high ratio of ASCII characters to national characters. The 1.5 multiplier might also be sufficient for most Asian language data. Even though many Asian characters require 3 bytes in UTF-8, the double-byte (DBCS) encodings that support those languages already require 2 bytes for each Asian character.

However, many languages that are represented by single-byte (SBCS) encodings use a high ratio of national characters in the text, which require 2 or 3 bytes in UTF-8. For example, Russian text primarily uses characters from the Cyrillic alphabet, which require 1 byte in the SBCS encoding WCYRILLIC (CP1251 and WINDOWS-1251). Since each Cyrillic character requires 2 bytes in UTF-8, the 1.5 multiplier might not be sufficient to prevent truncation, depending on the length of the variable.

If you need to change the expansion, you can specify either the CVPMULTIPLIER= or CVPBYTES= option in the LIBNAME statement. CVPMULTIPLIER= specifies a multiplier value that is applied to the length of the variable. CVPBYTES= adds a specific number of bytes to the length of the character columns.

Here are examples:

```plaintext
libname mylib 'path-to-SAS-files' cvpmultiplier=2.5;
libname mylib 'path-to-SAS-files' cvpbytes=5;
```

The following data tables can help you determine the additional storage requirements when saving your data as UTF-8.

**Table 17.38** shows the storage size increase that you can expect as you move data from the specified encoding to UTF-8.

**Table 17.2 Saving Data as UTF-8 and Possible Storage Size Increases**

<table>
<thead>
<tr>
<th>Source Encoding</th>
<th>Languages Supported</th>
<th>Storage Size Increase in UTF-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>English, Malay, and other supported languages</td>
<td>0%</td>
</tr>
<tr>
<td>ISO-8859-1 (LATIN1) and Windows cp1252 (WLATIN1)</td>
<td>Western European</td>
<td>10%</td>
</tr>
<tr>
<td>ISO-8859-7, plain text</td>
<td>Greek</td>
<td>90%</td>
</tr>
</tbody>
</table>
Table 17.39 shows how many bytes are needed in order to represent the characters for the languages in the list.

**Table 17.3** UTF-8 Character Length by Language

<table>
<thead>
<tr>
<th>Character Length</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>US_ASCII characters</td>
</tr>
<tr>
<td>2 bytes</td>
<td>East and West European, Baltic, Greek, Turkish, Cyrillic, Hebrew, Arabic, and other supported character sets</td>
</tr>
<tr>
<td>3 bytes</td>
<td>Chinese, Japanese, Korean (CJK), Thai, Indic, and certain control characters</td>
</tr>
<tr>
<td>4 bytes</td>
<td>Emoji characters, less common CJK characters, and various historic scripts</td>
</tr>
</tbody>
</table>

**Example: Using the CVP Engine**

The following example illustrates how to avoid character data truncation by using the CVP engine. The example uses a SAS data set named MYFILES.WLATIN2, which contains some national characters in the Wlatin2 encoding. The data set is created in SAS 9 in an SBCS session. This code sample is run in SAS9.

```sas
libname myfiles 'SAS Data-Library';
data myfiles.wlatin2 (encoding=wlatin2);
  var1='A';
  var2='Š';
  var3='ś';
  var4='ł';
;
proc print data=myfiles.wlatin2;
run;
```
Here is the PROC CONTENTS output for MYFILES.WLATIN2 that was run in SAS 9, which shows that the encoding is WLatin2 and the length for each character variable is 1 byte:

<table>
<thead>
<tr>
<th>Obs</th>
<th>var1</th>
<th>var2</th>
<th>var3</th>
<th>var4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Š</td>
<td>š</td>
<td>ł</td>
</tr>
</tbody>
</table>
## Output 17.1 PROC CONTENTS Output for MYFILES.WLATIN2

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>MYFILES.WLATIN2</th>
<th>Observations</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>4</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>07/28/2017 12:02:40</td>
<td>Observation Length</td>
<td>4</td>
</tr>
<tr>
<td>Last Modified</td>
<td>07/28/2017 12:02:40</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td>Compressed</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Data Set Type</td>
<td>Sorted</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>WINDOWS_64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>wlatin2 Central Europe (Windows)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Engine/Host Dependent Information

<table>
<thead>
<tr>
<th>Data Set Page Size</th>
<th>65536</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Data Set Pages</td>
<td>1</td>
</tr>
<tr>
<td>First Data Page</td>
<td>1</td>
</tr>
<tr>
<td>Max Obs per Page</td>
<td>15881</td>
</tr>
<tr>
<td>Obs in First Data Page</td>
<td>1</td>
</tr>
<tr>
<td>Number of Data Set Repairs</td>
<td>0</td>
</tr>
<tr>
<td>ExtendObsCounter</td>
<td>YES</td>
</tr>
<tr>
<td>Filename</td>
<td>C:\SAS Data-Library\wlatin2.sas7bdat</td>
</tr>
<tr>
<td>Release Created</td>
<td>9.0401M 5</td>
</tr>
<tr>
<td>Host Created</td>
<td>X64_7PRO</td>
</tr>
<tr>
<td>Owner Name</td>
<td>BUILTIN\Administrators</td>
</tr>
<tr>
<td>File Size</td>
<td>128KB</td>
</tr>
<tr>
<td>File Size (bytes)</td>
<td>131072</td>
</tr>
</tbody>
</table>

### Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>var1</td>
<td>Char</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>var2</td>
<td>Char</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>var3</td>
<td>Char</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>var4</td>
<td>Char</td>
<td>1</td>
</tr>
</tbody>
</table>
The following code is executed with the session encoding WLatin2.

```sas
options msglevel=i;
libname myfiles 'SAS Data-Library';
data myfiles.utf8 (encoding="utf-8");
   set myfiles.wlatin2;
run;
```

The DATA step requests a new data set named MYFILES.UTF8. It also requests that the data be read into the new data set in the UTF-8 encoding, which means that the data must be transcoded from WLatin2 to UTF-8. The request results in errors when the transcoding truncates the character data. The new data set MYFILES.UTF8 is created but does not contain any data.

**Example Code 17.1  SAS Log with Transcoding Error**

```sas
INFO: Data file MYFILES.UTF8.DATA is in a format 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.
ERROR: Some character data was lost during transcoding in the dataset MYFILES.UTF8. Either the data contains characters that are not representable in the new encoding or truncation occurred during transcoding.
NOTE: The step has been abnormally terminated.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 1 observations read from the data set MYFILES.WLATIN2.
WARNING: The data set MYFILES.UTF8 may be incomplete. When this step was stopped there were 0 observations and 4 variables.
```

The following code is executed and expands the variable lengths:

```sas
libname myfiles 'SAS Data-Library';
libname expand cvp 'SAS Data-Library' cvpbytes=2;
data myfiles.utf8 (encoding="utf-8");
   set expand.wlatin2;
run;
```

In this example, the CVP engine expands character variable lengths by adding 2 bytes to each length. The data is read into the new file in UTF-8 encoding by transcoding from WLatin2 to UTF-8. There is no data truncation due to the expanded character variable lengths, and the new data set is successfully created:
Example Code 17.2  SAS Log Output for MYFILES.UTF8

57     options msglevel=i;
58     libname myfiles 'C:\SAS Data-Library';

NOTE: Libref MYFILES was successfully assigned as follows:
   Engine:        V9
   Physical Name:  C:\SAS Data-Library
59     libname expand cvp 'C:\SAS Data-Library' cvpbytes=2;
WARNING: Libref EXPAND refers to the same physical library as MYFILES.
NOTE: Libref EXPAND was successfully assigned as follows:
   Engine:        CVP
   Physical Name: 'C:\SAS Data-Library'
60     data myfiles.utf8 (encoding="utf-8");
61     set expand.wlatin2;
INFO: Data file EXPAND.WLATIN2.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.
62     run;

NOTE: There were 1 observations read from the data set EXPAND.WLATIN2.
NOTE: The data set MYFILES.UTF8 has 1 observations and 4 variables.

Finally, here is PROC CONTENTS output for MYFILES.UTF8 showing that it is in the UTF-8 encoding and that the length of each character variable is 3:
### Output 17.2  PROC CONTENTS Output for MYFILES.UTF8

#### Data Set Name
- **MYFILES.UTF8**
- **Observations**: 1

#### Member Type
- **DATA**
- **Variables**: 4

#### Engine
- **V9**
- **Indexes**: 0

#### Created
- **07/28/2017 12:02:41**
- **Observation Length**: 12

#### Last Modified
- **07/28/2017 12:02:41**
- **Deleted Observations**: 0

#### Protection
- **Compressed**
- **NO**

#### Data Set Type
- **Sorted**
- **NO**

#### Label

#### Data Representation
- **WINDOWS_64**

#### Encoding
- **utf-8 Unicode (UTF-8)**

### Engine/Host Dependent Information

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>65536</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Data Set Pages</td>
<td>1</td>
</tr>
<tr>
<td>First Data Page</td>
<td>1</td>
</tr>
<tr>
<td>Max Obs per Page</td>
<td>5403</td>
</tr>
<tr>
<td>Obs in First Data Page</td>
<td>1</td>
</tr>
<tr>
<td>Number of Data Set Repairs</td>
<td>0</td>
</tr>
<tr>
<td>ExtendObsCounter</td>
<td>YES</td>
</tr>
<tr>
<td>Filename</td>
<td>C:\SAS Data-Library\utf8.sas7bdat</td>
</tr>
<tr>
<td>Release Created</td>
<td>9.0401M5</td>
</tr>
<tr>
<td>Host Created</td>
<td>X64_7PRO</td>
</tr>
<tr>
<td>Owner Name</td>
<td>BUILTIN\Administrators</td>
</tr>
<tr>
<td>File Size</td>
<td>128KB</td>
</tr>
<tr>
<td>File Size (bytes)</td>
<td>131072</td>
</tr>
</tbody>
</table>

### Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>var1</td>
<td>Char</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>var2</td>
<td>Char</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>var3</td>
<td>Char</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>var4</td>
<td>Char</td>
<td>3</td>
</tr>
</tbody>
</table>
ENCODING= Option Statement

Overrides and transcodes the encoding for input or output processing of external files.

Valid in:

- %INCLUDE statement
- FILE statement
- FILENAME statement
- EMAIL (SMTP) Access Method
- INFILE statement
- ODS statements
- FILE command
- INCLUDE command

Category: Data Access

Syntax

ENCODING = 'encoding-value'

Optional Argument

ENCODING = 'encoding-value'

specifies the encoding to use for reading, writing, copying, or saving an external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read, write, copy, or save data using an external file, SAS transcodes the data from the session encoding to the specified encoding.

For details, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

Default: SAS uses the current session encoding.

Details

The following table provides information about how the ENCODING option is used with the corresponding statement:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%INCLUDE statement</td>
<td>reads SAS statements and data lines from the specified source file (not supported under z/OS).</td>
</tr>
<tr>
<td>FILE statement</td>
<td>writes to an external file.</td>
</tr>
<tr>
<td>FILENAME statement</td>
<td>reads from or writes to an external file.</td>
</tr>
<tr>
<td>FILENAME statement, EMAIL (SMTP) Access Method</td>
<td>sends electronic mail programmatically.</td>
</tr>
<tr>
<td>INFILE statement</td>
<td>reads from an external file.</td>
</tr>
<tr>
<td>ODS statements</td>
<td>controls features of the Output Delivery System that are used to generate, store, or reproduce SAS procedure and DATA step output.</td>
</tr>
</tbody>
</table>
FILE command: saves the contents of a window to an external file.

INCLUDE command: Copies an external file into the current window.

Some encodings use a Byte-Order Mark (BOM). The BOM is generated when the encoding is specified. For the UTF-8 encoding, you must specify encoding=utf-8 on the filename and file DATA step statements in order for the BOM to be generated.

Examples:

Example 1: Using the FILE Statement to Specify an Encoding for Writing to an External File

This example creates an external file from a SAS data set. The current session encoding is Wlatin1, but the external file's encoding needs to be UTF-8. By default, SAS writes the external file using the current session encoding.

To specify what encoding to use for writing data to the external file, specify the ENCODING= option:

```sas
libname myfiles 'SAS data-library';
filename outfile 'external-file';
data _null_
   set myfiles.cars;
   file outfile encoding="utf-8";
   put Make Model Year;
r
run;
```

When you tell SAS that the external file is to be in UTF-8 encoding, SAS then transcodes the data from Wlatin1 to the specified UTF-8 encoding.

Example 2: Using the FILENAME Statement to Specify an Encoding for Reading an External File

This example creates a SAS data set from an external file. The external file is in UTF-8 character-set encoding, and the current SAS session is in the Wlatin1 encoding. By default, SAS assumes that an external file is in the same encoding as the session encoding, which causes the character data to be written to the new SAS data set incorrectly.

To specify which encoding to use when reading the external file, specify the ENCODING= option:

```sas
libname myfiles 'SAS data-library';
filename extfile 'external-file' encoding="utf-8";
data myfiles.unicode;
infile extfile;
input Make $ Model $ Year;
r
run;
```

When you specify that the external file is in UTF-8, SAS then transcodes the external file from UTF-8 to the current session encoding when writing to the new SAS data set. Therefore, the data is written to the new data set correctly in Wlatin1.
Example 3: Using the FILENAME Statement to Specify an Encoding for Writing to an External File

This example creates an external file from a SAS data set. By default, SAS writes the external file using the current session encoding. The current session encoding is Wlatin1, but the external file’s encoding needs to be UTF-8.

To specify which encoding to use when writing data to the external file, specify the ENCODING= option:

```sas
libname myfiles 'SAS data-library';
filename outfile 'external-file' encoding="utf-8";
data _null_;  
    set myfiles.cars;  
    file outfile;  
    put Make Model Year;
run;
```

When you specify that the external file is to be in UTF-8 encoding, SAS then transcodes the data from Wlatin1 to the specified UTF-8 encoding when writing to the external file.

Example 4: Changing Encoding for Message Body and Attachment

This example illustrates how to change text encoding for the message body as well as for the attachment.

```sas
filename mymail email 'Joe.Developer@sas.com';
data _null_;  
    file mymail
        subject='Text Encoding'
        encoding=greek
        attach=('C:\My Files\Test.out'
            content_type='text/plain'
            encoding='ebcdic1047'
            outencoding='latin1');
run;
```

In the program, the following occurs:

- The ENCODING= email option specifies that the message body is encoded to Greek (ISO) before being sent.
- For the ATTACH= email option, the attachment option ENCODING= specifies the encoding of the attachment that is read into SAS, which is Western (EBCDIC).
- Because SMTP and other email interfaces do not support EBCDIC, the attachment option OUTENCODING= converts the attachment to Western (ISO) before sending it.

Example 5: Using the INFILE= Statement to Specify an Encoding for Reading from an External File

This example creates a SAS data set from an external file. The external file’s encoding is in UTF-8, and the current SAS session encoding is Wlatin1. By default, SAS assumes that the external file is in the same encoding as the session encoding, which causes the character data to be written to the new SAS data set incorrectly.
To specify which encoding to use when reading the external file, specify the ENCODING= option:

```
libname myfiles 'SAS data-library';
filename extfile 'external-file';
data myfiles.unicode;
   infile extfile encoding="utf-8";
   input Make $ Model $ Year;
run;
```

When you specify that the external file is in UTF-8, SAS then transcodes the external file from UTF-8 to the current session encoding when writing to the new SAS data set. Therefore, the data is written to the new data set correctly in WLatin1.

See Also

- "%INCLUDE Macro Statement: UNIX" in SAS Companion for UNIX Environments
- "%INCLUDE Statement: Windows" in SAS Companion for Windows
- "FILE Statement" in SAS DATA Step Statements: Reference
- "INFILE Statement" in SAS DATA Step Statements: Reference

 Commands:
- "FILE Command: z/OS" in SAS Companion for z/OS
- "FILE Command: UNIX" in SAS Companion for UNIX Environments
- "FILE Command: Windows" in SAS Companion for Windows
- "INCLUDE Command: z/OS" in SAS Companion for z/OS
- "INCLUDE Command: Windows" in SAS Companion for Windows

INENCODING= Statement

OUTENCODING= Options Statement

Overrides and changes the encoding when reading or writing SAS data sets in the SAS library.

Valid in: LIBNAME statement
Category: Data Access

Syntax

```
INENCODING= ANY | ASCIIANY | EBCDICANY | encoding-value
OUTENCODING= ANY | ASCIIANY | EBCDICANY | encoding-value
```
Syntax Description

**ANY**
specifies no transcoding between ASCII and EBCDIC encodings.

NOTE: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

**ASCIIANY**
specifies that no transcoding occurs, assuming that the mixed encodings are ASCII encodings.

**EBCDICANY**
specifies that no transcoding occurs, assuming that the mixed encodings are EBCDIC encodings.

**encoding-value**
specifies an encoding value. For a list of encoding values, see Chapter 23, “Encoding Values in SAS Language Elements,” on page 829.

Details

The INENCODING= option is used to read SAS data sets in the SAS library. The OUTENCODING= option is used to write SAS data sets in the SAS library.

The INENCODING= value or the OUTENCODING= value is written to the SAS log when you use the LIST argument.

INENCODING= and OUTENCODING= are most appropriate when using an existing library that contains mixed encodings. To read a library that contains mixed encodings, you can set INENCODING= to ASCIIANY or EBCDICANY. To write a separate data set, you can use OUTENCODING= to specify a specific encoding, which is applied to the data set when it is created.

Comparisons

- Session encoding is specified using the ENCODING= system option.
- You can specify the encoding for reading data sets in a SAS library by using the LIBNAME statement INENCODING= option for input files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.
- You can specify the encoding for writing data sets to a SAS library by using the LIBNAME statement OUTENCODING= option for output files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.
- For the COPY procedure, the default CLONE option uses the encoding attribute of the input data set instead of the encoding value specified on the OUTENCODING= option. For more information about CLONE and NOCLONE, see COPY Statement.

Note: This interaction does not apply when using SAS/CONNECT or SAS/SHARE.
See Also

- “Overview: Encoding for NLS” on page 9

System Options:

- “ENCODING System Option: UNIX, Windows, and z/OS” on page 694
- “LOCALE System Option” on page 698

Data Set Options:

- “ENCODING= Data Set Option” on page 69

ODSCHARSET= Option Statement

Specifies the character set to be generated in the META declaration for the output.

Valid in: LIBNAME statement for the XML V2 engine

Category: Data Access

Syntax

**ODSCHARSET=character-set;**

Required Argument

**character-set**

For the LIBNAME statement for the XML engine, specifies the character set to use in the ENCODING= attribute.

An example of an encoding is ISO-8859-1. Official character sets for use on the Internet are registered by IANA (Internet Assigned Numbers Authority). IANA is the central registry for various Internet protocol parameters, such as port, protocol and enterprise numbers, options, codes, and types. For a complete list of character-set values, see [www.unicode.org/reports/tr22/index.html](http://www.unicode.org/reports/tr22/index.html) and [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets).

A character set is like an encoding-value in this context. However, character set is the term that is used to identify an encoding that is suitable for use on the Internet.

Details

An XML declaration is not required in all XML documents. Such a declaration is required only when the character encoding of the document is other than the default UTF-8 or UTF-16 and no encoding was determined by a higher-level protocol.

The ODSCHARSET option, in the LIBNAME statement for the XML engine, specifies the character set to use for generating an output XML document.
See Also

Conceptual Information:
- Chapter 3, “Encoding for NLS,” on page 9

Statements:
- SAS XMLV2 and XML LIBNAME Engines: User’s Guide

ODSTRANTAB= Option Statement

Specifies the translation table to use when transcoding an XML document for an output file.

Valid in: the LIBNAME statement for the XML V2engine
Category: Data Access

Syntax
TRANTAB = 'translation-table'

Optional Argument

translation-table
specifies the translation table to use for the output file. The translation table is an encoding method that maps characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) in the character set to numeric values. An example of a translation table is one that converts characters from EBCDIC to ASCII-ISO. The table-name can be any translation table that SAS provides, or any user-defined translation table. The value must be the name of a SAS catalog entry in either the SASUSER.PROFILE catalog or the SASHELP.HOST catalog.

Details

For SAS 9.2, using the ODSTRANTAB= option in the LIBNAME statement for the XML Engine is supported for backward compatibility. The preferred method for specifying an encoding is to use the LOCALE= system option.

See Also

Conceptual Information:
- “Transcoding and Translation Tables” on page 32
- Conceptual discussion of Chapter 2, “Locale for NLS,” on page 5

System Options:
- “TRANTAB= System Option” on page 711
- “LOCALE System Option” on page 698
TRANSCODE= Column Modifier on PROC SQL

Specifies whether values can be transcoded for character columns.

Valid in: Column modifier component in the SQL Procedure

Syntax

TRANSCODE=YES|NO

Required Argument

TRANSCODE=YES | NO

for character columns, specifies whether values can be transcoded. Use TRANSCODE=NO to suppress transcoding. Note that when you create a table using the CREATE TABLE AS statement, the transcoding attribute for a particular character column in the created table is the same as it is in the source table unless you change it with the TRANSCODE= column modifier.

Default YES

Restriction

Suppression of transcoding is not supported for the V6TAPE engine.

See Also

Conceptual Information:

- Chapter 4, “Transcoding for NLS,” on page 31

RENCODING= Option Statement

Specifies the ASCII-based or EBCDIC-based encoding to use for transcoding data for a SAS/SHARE server session that is using an EBCDICANY or ASCIIANY session encoding.

Valid in: LIBNAME statement for SAS/SHARE only

Category: Data Access

Note: The RENCODING= option in the LIBNAME statement is relevant only if using a SAS/SHARE server that has a session encoding set to EBCDICANY or ASCIIANY to preserve a mixed-encoding computing environment.
See: LIBNAME statement in SAS/SHARE User’s Guide

Syntax

RENCODING=ASCII-encoding-value | EBCDIC-encoding-value

Syntax Description

ASCII-encoding-value

For a list of valid values for ASCII encodings for UNIX and Windows, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

EBCDIC-encoding-value

For a list of valid values for EBCDIC encodings for z/OS, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

Details

If you use SAS/SHARE in a mixed-encoding environment (for example, SAS/SHARE client sessions using incompatible encodings such as Latin1 and Latin2), you can set the following options:

- in the SAS/SHARE server session, set the SAS system option ENCODING=EBCDICANY or ENCODING=ASCIIANY
- in the SAS/SHARE client session, set the RENCODING= option in the LIBNAME statement(s) under these conditions:
  - a client session that uses an ASCII-based encoding accesses an EBCDICANY server
  - a client session that uses an EBCDIC-based encoding accesses an ASCIIANY server.

The RENCODING= option enables SAS/SHARE clients to specify which encoding to assume the server’s data is in when transcoding to or from the client session encoding.

For SAS 9, if you are processing data in a SAS/SHARE client/server session from more than one SBCS or DBCS encoding, you are advised to use the UTF8 encoding. For more information about Unicode servers that run the UTF8 session encoding, go to http://rnd.sas.com/sites/i18n/i18ndocs/i18nsupport/Pages/SAS%20Technical%20Papers.aspx and search for Processing Multilingual Data with the SAS® 9.2 Unicode Server and Multilingual Computing with SAS® 9.4.

Comparisons

In SAS 9, you can maintain multilingual data that contains characters from more than one traditional SBCS or DBCS encoding in a SAS data set by using a UTF8 encoding. To share Update access to that data using SAS/SHARE, you must also run the SAS/SHARE server using a session encoding of UTF8. SAS transcodes the data to the client encoding if necessary.

Prior to SAS 9, if a SAS/SHARE client and a SAS/SHARE server ran on common architectures (for example, the client and server ran on UNIX machines), there was no automatic transcoding of character data. It was possible to build applications that accessed data sets in different EBCDIC or ASCII encodings within a single SAS/SHARE server, or that accessed data sets in mixed different encodings within
a single data set. This method was very uncommon and required careful programming to set up transcoding tables from clients that ran in different operating environments.

The following steps describe how you can maintain mixed encoding in SAS 9, if necessary.

- The SAS/SHARE server must run by using a session encoding of EBCDICANY for mixed-EBCDIC encodings or ASCIIANY for mixed-ASCII encodings.

  This restores the behavior of Version 8 and earlier releases and prevent the automatic character transcoding between different client and server encodings in the same EBCDIC or ASCII family. That is, no transcoding occurs under these circumstances:
  - if the client session encoding is an EBCDIC encoding and the server session encoding is EBCDICANY
  - if the client session encoding is an ASCII encoding and the server session encoding is ASCIIANY.

- A SAS/SHARE client that does not share the same encoding family as an ASCIIANY or EBCDICANY server can control the necessary transcoding by using an RENCODING= option on the first LIBNAME statement that accesses the server.

  For example, an ASCII client that runs in a Polish locale could access a z/OS EBCDICANY server and specify RENCODING=EBCDIC870 to access data that the client knows contains Polish-encoded data. Another ASCII client that runs in a German locale could access the same z/OS EBCDICANY server and specify RENCODING=EBCDIC1141 to access data that the client knows contains German data. Similarly, EBCDIC clients that access an ASCIIANY server can specify the precise ASCII encoding of the data that they are accessing by using the RENCODING= option in the LIBNAME statement.

See Also

- Conceptual Information:
  - “Overview of Transcoding” on page 31

---

**TRANSCODE= Option Statement**

Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and length with one or more variables) that indicates whether character variables are to be transcoded.

Valid in: the ATTRIB statement in a DATA step

Category: Information

Type: Declarative

See: ATTRIB Statement under Windows UNIX z/OS in the documentation for your operating environment.
Syntax

ATTRIB variable-list(s) attribute-list(s) ;

Required Arguments

**variable-list**

names the variables that you want to associate with the attributes.

*Tip* List the variables in any form that SAS allows.

**attribute-list**

specifies one or more attributes to assign to **variable-list**. Multiple attributes can be specified in the ATTRIB statement. For a complete list of attributes, see the “ATTRIB Statement” in *SAS DATA Step Statements: Reference*.

**TRANSCODE= YES | NO**

Specifies whether to transcode character variables. Use TRANSCODE=NO to suppress transcoding. For more information, see “Overview of Transcoding” on page 31.

*Default* YES

*Restriction* The TRANSCODE=NO attribute is not supported by some SAS Workspace Server clients. Variables with TRANSCODE=NO are not returned in SAS 9.4. Prior to SAS 9.4, variables with TRANSCODE=NO are transcoded. Prior releases of SAS cannot access a SAS 9.4 data set that contains a variable with a TRANSCODE=NO attribute.

*Interactions* You can use the VTRANSCODE and VTRANSCODEX functions to return whether transcoding is on or off for a character variable.

If the TRANSCODE= attribute is set to NO for any character variable in a data set, PROC CONTENTS prints a transcode column that contains the TRANSCODE= value for each variable in the data set. If all variables in the data set are set to the default TRANSCODE= value (YES), no transcode column is printed.

**Examples:**

**Example 1: Using the TRANSCODE= Option with the SET Statement**

When you use the SET statement to create a data set from several data sets, SAS makes the TRANSCODE= attribute of the variable in the output data set equal to the TRANSCODE= value of the variable in the first data set. In this example, the variable Z's TRANSCODE= attribute in data set A is NO because B is the first data set and Z's TRANSCODE= attribute in data set B is NO.

```sas
data b;
   length z $4;
   z = 'ice';
   attrib z transcode = NO;
data c;
```
Example 2: Using the TRANSCODE= Option with the MERGE Statement

When you use the MERGE statement to create a data set from several data sets, SAS makes the TRANSCODE= attribute of the variable in the output data set equal to the TRANSCODE= value of the variable in the first data set. In this example, the variable Z's TRANSCODE= attribute in data set A is YES because C is the first data set and Z's TRANSCODE= attribute in data set C is YES.

```
data b;
  length z $4;
  z = 'ice';
  attrib z transcode = NO;
data c;
  length z $4;
  z = 'snow';
  attrib z transcode = YES;
data a;
  merge c b;
  /* Check transcode setting for variable Z */
  rc1 = vtranscode(z);
  put rc1=;
run;
```

Note: The TRANSCODE= attribute is set when the variable is first seen on an input data set or in an ATTRIB TRANSCODE= statement. If a SET or MERGE statement comes before an ATTRIB TRANSCODE= statement and the TRANSCODE= attribute contradicts the SET statement, an error message occurs.

See Also

Functions:
- "VTRANSCODE Function" on page 514
- "VTRANSCODEX Function" on page 515

TRANTAB= Option Statement

Specifies the translation table to use when you are transcoding character data.
Valid in: ODS MARKUP statement and ODS RTF statement
Category: ODS: Third-Party Formatted

Syntax

TRANTAB = (translation-table)

Optional Argument

translation-table
specifies the translation table to use for the output file. The translation table is an encoding method that maps characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) in the character set to numeric values. An example of a translation table is one that converts characters from EBCDIC to ASCII-ISO. The table-name can be any translation table that SAS provides, or any user-defined translation table. The value must be the name of a SAS catalog entry in either the SASUSER.PROFILE catalog or the SASHELP.HOST catalog.

Details

In SAS 9.4, the TRANTAB= option in the ODS MARKUP statement is supported for backward compatibility. The ENCODING= option is preferred when specifying the encoding.

See Also

Conceptual Information:
- “Transcoding and Translation Tables” on page 32

Procedures:
- Chapter 20, “TRANTAB Procedure,” on page 783

Statements:

XMLENCODING= Option Statement

Overrides the encoding of an XML document to import or export an external document.

Valid in: LIBNAME statement for the XML engine
Category: Data Access
Syntax

XMLENCODING = 'encoding-value'

Details

The LIBNAME statement for the XML engine, associates a SAS libref with an XML document to import or export an external document.

Comparisons

Options

encoding-value

specifies the encoding to use when you read, write, copy, or save an external file. The value for XMLENCODING= indicates that the external file has a different encoding from the current session encoding.

For details, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 829.

The default for encoding-value is the current session encoding.

See Also

Statements:

- SAS XMLV2 and XML LIBNAME Engines: User’s Guide

TRANTAB Statement

Specifies the translation table to use when you transcode character data in order to export or transfer a SAS file.

Valid in: CPORT Procedure, UPLOAD procedure, DOWNLOAD procedure

Restriction: You can specify only one translation table per TRANTAB statement. To specify additional translation tables, use additional TRANTAB statements.

Interaction: The TRANTAB statement specifies a customized translation table (for example, to map an EBCDIC character to an ASCII character) to apply to the character set in the SAS file that is being exported or transferred. The TRANTAB= system option specifies a translation table to use for the SAS session, including file transfers.

Syntax

TRANTAB NAME=translation-table-name <TYPE=(etype-list) <OPT=DISP | SRC | (DISP SRC)> > ;
Required Argument

**NAME=translation-table-name**

specifies the name of the translation table to apply to the SAS catalog that you want to export (PROC CPORT) or transfer (PROC UPLOAD or PROC DOWNLOAD). The *translation-table-name* that you specify as the name of a catalog entry in either your SASUSER.PROFILE catalog or the SASHELP.HOST catalog. The SASUSER.PROFILE catalog is searched first, and then the SASHELP.HOST catalog is searched.

In most cases, the default translation table is the correct one to use, but you might need to apply additional translation tables if, for example, your application requires different national language characters.

You can specify a translation table other than the default in two ways:

- To specify a translation table for an invocation of the procedure, use the TRANTAB statement in the procedure, as appropriate.
- To specify a translation table for your entire SAS session or job (including all file exports or transfers), use the TRANTAB= system option.

Optional Arguments

**TYPE=(etype-list)**

applies the translation table only to the entries with the type or types that you specify. The *etype-list* can be one or more entry types. Examples of catalog entry types include DATA and FORMAT. If *etype-list* is a simple entry type, omit the parentheses.

By default, the UPLOAD, DOWNLOAD, and CPORT procedures apply the translation table to all specified catalog entries.

**OPT=DISP | SRC ( | )**

- **OPT=DISP** applies the translation table only to the specified catalog entries, which produce window displays.
- **OPT=SRC** applies the translation table only to the specified catalog entries that are of the type SOURCE.
- **OPT=(DISP SRC)** applies the translation table only to the specified catalog entries that either produce window displays or are of type SOURCE.

If you do not specify the OPT= option, the UPLOAD or DOWNLOAD procedure applies the translation table to all of the entries in the catalog that you specify.

**Default** PROC CPORT, PROC UPLOAD, and PROC DOWNLOAD apply the translation table to all entries and data sets in the specified catalog.

Details

Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on direct use of translation tables. SAS 9.3 supports the TRANTAB statement for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. For more information, see TS-639, Data Conversion Issues in V6–V8. This technical support note provides information for customers using non-English languages [http://support.sas.com/techsup/technote/ts639.pdf](http://support.sas.com/techsup/technote/ts639.pdf)
PROC CPORT is used when you transfer a SAS file across a network. PROC UPLOAD and PROC DOWNLOAD are used when you transfer a SAS file across a network.

You must specify the INCAT= and OUTCAT= options in the PROC UPLOAD or PROC DOWNLOAD statement when using the TRANTAB statement.

Examples:

Example 1

The information that follows applies to procedure features:

- PROC CPORT statement option: FILE=
- TRANTAB statement option: TYPE=

This example shows how to apply a customized translation table to the transport file before PROC CPORT exports it. For this example, assume that you have already created a customized translation table called TTABLE1.

Example 2: Program

Assign library references. The LIBNAME and FILENAME statements assign a libref for the source library and a fileref for the transport file, respectively.

```sas
libname source '\\sashq\root\pub\pubdoc\doc\901\authoring\proc\miscsrc\sasfiles\cport';
filename tranfile 'trans3';
proc trantab table=ascii;
save table=ttable1;
```

libname source 'SAS data-library';
filename tranfile 'transport-file' host-options-for-file-characteristics;

Apply the translation specifics. The TRANTAB statement applies the translation that you specify with the customized translation table TTABLE1. TYPE= limits the translation to FORMAT entries.

```sas
proc cport catalog=source.formats file=tranfile;
  trantab name=ttable1 type=(format);
run;
```

Example 3: SAS Log

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FORMATS
NOTE: The catalog has 2 entries and its maximum logical record length is 104.
NOTE: Entry REVENUE.FORMAT has been transported.
NOTE: Entry DEPT.FORMATC has been transported.
```
See Also

Conceptual Information:
- Chapter 4, “Transcoding for NLS,” on page 31

System Options:
- “TRANTAB= System Option” on page 711

Procedures:
- Chapter 20, “TRANTAB Procedure,” on page 783
- “CPORT Procedure” in Base SAS Procedures Guide
- “UPLOAD Procedure” in SAS/CONNECT User’s Guide
- “DOWNLOAD Procedure” in SAS/CONNECT User’s Guide
PART 10

Procedures for NLS

Chapter 18

DBCSTAB Procedure .......................................................... 761

Chapter 19

LOCALEDATA Procedure .................................................. 767

Chapter 20

TRANTAB Procedure .......................................................... 783
DBCSTAB Procedure

Overview: DBCSTAB Procedure

What Does the DBCSTAB Procedure Do?

The DBCSTAB procedure produces conversion tables for the double-byte character sets (DBCSs) that SAS supports.

Use the DBCSTAB procedure to modify an existing DBCS table when

- the DBCS encoding system that you are using is not supported by SAS
- the DBCS encoding system that you are using has a nonstandard translation table

You might be more likely to use the DBCSTAB procedure when a valid DBCSTYPE= value is not available. These values depend on the operating environment. In such cases, you can use the DBCSTAB procedure to modify a similar translation table, and then you can specify the use of the new table with the TRANTAB option.

Syntax: DBCSTAB Procedure

```
PROC DBCSTAB TABLE=table-name
   <BASETYPE=base-type> <CATALOG=libref.catalog-name>
   <DATA=libref.table-name> <DBCSLANG=language>
```

PROC DBCSTAB Statement

Produces conversion tables for double-byte character sets.

Syntax

PROC DBCSTAB TABLE=table-name <options>;

Required Argument

TABLE=table-name
specifies the name of the double-byte code table to produce. This table name becomes an entry of type DBCSTAB in the catalog that is specified with the CATALOG= option. By default, the catalog name is Sasuser.DBCS.

Alias  NAME=, N=

Optional Arguments

BASETYPE=base-type
specifies a base type for the double-byte code table conversion. If you use this option, you reduce the number of tables that are produced.

If you specify BASETYPE=, then all double-byte codes are first converted to the base code, and then they are converted to the required code. If you have $n$ codes, then there are $n(n-1)$ conversions that must be made.

Alias  BTYPE=

CATALOG=<libref.>catalog-name
specifies the name of the catalog in which the table is stored. If the catalog does not exist, it is created.

Default  Sasuser.DBCS

DATA=<libref.>table-name
specifies the data for producing the double-byte code table. Several double-byte character variables are required to produce the table. Use variable names that are equivalent to the value of the DBCSTYPE= system option and recognized by the KCVT function.

DBCSLANG=language
specifies the language that the double-byte code table uses. The value of this option should match the value of the DBCSLANG= system option.

Alias  DBLANG=

DESC='description'
specifies a text string to put in the DESCRIPTION field for the entry.
FORCE produces the conversion tables, even if errors are present.

VERIFY checks the data range of the input table based on the code. This option checks for invalid double-byte code.

VERBOSE prints the statistics detail when building DBCS tables.

---

Examples: DBCSTAB Procedure

Example 1: Creating a Conversion Table with the DBCSTAB Procedure

Features:

PROC DBCSTAB statement options:
- CATALOG=
- DBLANG=
- BASETYPE=
- VERIFY

The following example creates a Japanese translation table called CUSTAB and demonstrates how the TRANTAB= option specifies the new translation table. The DBCS=, DBCSLANG=, and DBCSTYPE= options are specified at start-up.

Program

```sas
proc dbcstab
   /* name of the new translate table */
   name=custtab
   /* based on pcibm encoding */
   basetype=pcms
   /* data to create the new table */
   data=trantab
   /* japanese language */
   dbcslang=japanese
   /* catalog descriptor */
   desc='Modified Japanese Trantab'
   /* where the table is stored */
   catalog=sasuser.dbcs
   /* checks for invalid DBCS in the new data */
   verify;
run;
```

Creating a Conversion Table

invoke Japanese SAS with Shift_JIS encoding.
data trantab;
  pcms='8342'x; dec='b9b3'x;
run;
proc dbcstab;
  name=custtab;
  basetype=pcms;
  data=trantab;
  dbcslang=japanese;
  desc='Modified Japanese Trantab';
  catalog=sasuser.dbcs;
  verify;
run;
options trantab={,,,,,,,,,custtab};

Program Description

Set the SAS environment.
  invoke Japanese SAS with Shift_JIS encoding.

Create the TRANTAB data set.
  data trantab;

Define the pcms and dec parameters and run the code.
  pcms='8342'x; dec='b9b3'x;
run;

Invoke the DBCSTAB procedure.
  proc dbcstab;

Assign a name to a translate table.
  name=custtab;

Apply pcibm encoding.
  basetype=pcms;

Access data from the TranTab data set.
  data=trantab;

Set the language to Japanese.
  dbcslang=japanese;

Set the catalog descriptor to Modified Japanese Translation table.
  desc='Modified Japanese Trantab';

Set the table’s location.
  catalog=sasuser.dbcs;

Check for invalid DBCS data and then process the program.
  verify;
run;
Select the translate table using the TRANTAB= option. Translate tables are used for DBCS conversion with SAS/CONNECT, PROC CPORT and PROC CIMPORT, and the DATA step function KCVT. The TRANTAB= option can specify DBCS translate tables. For SAS 8.2 and earlier, the ninth argument specified the DBCS system table. However, for SAS 9 and later, instead of using the ninth argument, SAS uses a system table that is contained in a module. Japanese, Korean, Chinese, and Taiwanese are acceptable for the SYSTAB name. The 10th argument specifies the DBCS user table:

    options trantab=(,,,,,,,,custtab);

Example 2: Producing Japanese Conversion Tables with the DBCSTAB Procedure

Features:

    PROC DBCSTAB statement options:
      TABLE=
      DATA=
      DBLANG=
      BASETYPE=
      VERIFY

Program

data ja_jpn;
  length ibm jis euc pcibm $2.;
  ibm='4040'x;
  jis='2121'x;
  euc='a1a1'x;
  pcibm='8140'x;
run;
proc dbcstab;
  table=japanese;
  data=ja_jpn;
  dblang=japanese;
  basetype=jis;
  verify;
run;

Program Description

Define the ja_jpn data set.

    data ja_jpn;

Set the length of the specified encodings to 2.

    length ibm jis euc pcibm $2.;
Assign the value 4040 to the ibm encoding.
    ibm='4040'x;

Assign the value 2121 to the jis encoding.
    jis='2121'x;

Assign the value a1a1 to the euc encoding.
    euc='a1a1'x;

Assign the value 8140 to the pcibm encoding and run the code.
    pcibm='8140'x;
    run;

Invoke the DBCSTAB procedure.
    proc dbcstab;

Define the table with the name, Japanese.
    table=japanese;

Access the Ja_Jpn data set.
    data=ja_jpn;

Specify Japanese for the language.
    dblang=japanese;

Specify the jis encoding.
    basetype=jis;

Check for invalid DBCS data and then process the program.
    verify;
    run;

SAS Log

1   proc dbcstab
2   table=ja_jpn
3   data=work.ja_jpn
4   dblang=japanese
5   basetype=jis
6   verify;
7   run;

NOTE: Base table for JIS created.
NOTE: IBM table for JIS created.
NOTE: PCIBM table for JIS created.
NOTE: EUC table for JIS created.
NOTE: Base table for IBM created.
NOTE: JIS table for IBM created.
NOTE: Base table for PCIBM created.
NOTE: JIS table for PCIBM created.
NOTE: Base table for EUC created.
NOTE: JIS table for EUC created.
NOTE: 10 DBCS tables are generated. Each table has 1 DBCS characters.
NOTE: Each table is 2 bytes in size.
NOTE: Required table memory size is 612.
NOTE: There were 1 observations read from the data set WORK.JA_JPN.
Overview: LOCALEDATA Procedure

What Does the LOCALEDATA Procedure Do?

The LOCALEDATA procedure lets you customize data. You can view, open, modify, and store the customized locale data.

Syntax: LOCALEDATA Procedure

PROC LOCALEDATA;
    LOAD SASLOCALE | REGISTRY;
    MODIFY key=key-name value=key-value | category=category-name value=locale;
    CONTENTS _ALL_ | key-name | category-name;
    SAVE REGISTRY < _ALL_ | key-name | category-name syntax=SAS>;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC LOCALEDATA</td>
<td>Specifies the source of customized locale data.</td>
</tr>
<tr>
<td>Statement</td>
<td>Task</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>LOAD</td>
<td>Specifies how to customize the locale data by locale element key or by locale category.</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>Specifies what locale element’s values are discarded.</td>
</tr>
<tr>
<td>SAVE</td>
<td>Stores the customized locale data in the SAS registry.</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Customizes the locale data by locale element key or by locale category.</td>
</tr>
</tbody>
</table>

**PROC LOCALEDATA Statement**

Specifies the source of customized locale data.

**Syntax**

```
PROC LOCALEDATA;
```

**LOAD Statement**

Specifies which source of locale data is loaded for customization.

Default: Registry

**Syntax**

```
LOAD SASLOCALE | REGISTRY;
```

**Required Arguments**

- **SASLOCALE**
  - loads the locale data from the factory setting default locale database.
- **REGISTRY**
  - loads the locale data from the SAS registry. This locale data could be customized locale data that has been stored in the SAS registry.

**MODIFY Statement**

Customizes the locale data by locale element key or by locale category.
Syntax

\[ \text{MODIFY } \text{key}=\text{name} \text{ value}=\text{value} \mid \text{category}=\text{category-name} \text{ value}=\text{locale} \]

Required Arguments

**key=\text{key-name} value=\text{key-value}**
- customizes single locale element values.

**category=\text{category-name} value=\text{locale}**
- customizes all locale elements in a category. You can select one of the following categories:
  - \( \text{LC\_NUMERIC} \)
  - \( \text{LC\_MONETARY} \)
  - \( \text{LC\_TIME} \)

The following table shows the element keys that you can modify.

The value of the locale element key must be less than the maximum length value.

The following values can be used for the type:

<table>
<thead>
<tr>
<th>Locale Element Key</th>
<th>Max Length</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATESTYLE</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PAPERSIZE</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTITLE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTEXT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMFONT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SORTSEQ</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>MESSAGES</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATE_TIME</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_TIME</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>------</td>
<td>----------------</td>
</tr>
<tr>
<td>FORMATNAME_NUMERIC</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_PERCENT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SANSSERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_Cursive</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_Fantasy</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_MONOSPACe</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BRUSH</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SWISS</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ITALIC</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DATE_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_AMPm_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_WEEK_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_WEEK_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME_AMPm_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>DATE_WEEK_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_WEEK_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYMM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYMM_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>ABMON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ABDAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>AM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>PM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>INT_CURRENCY_SYMBOL</td>
<td>3</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>CURRENCY_SYMBOL</td>
<td>32</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_DECIMAL_POINT</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_THOUSANDS_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>MON_POSITIVE_SIGNED</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_NEGATIVE_SIGNED</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_INT_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>DECIMAL_POINT</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>THOUSANDS_SEP</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>NEGATIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
</tbody>
</table>
CONTENTS Statement
Displays the value of the specified locale element key.

Syntax
CONTENTS _ALL_ | locale-element-key | category-name

Required Arguments
_ALL_
   displays the value of all locale element keys.

locale-element-key
   displays the value of the specified locale element key.

category-name
   displays the value of locale elements for the specified category.

SAVE Statement
Stores the customized locale data in the SAS registry.

Syntax
SAVE REGISTRY < _ALL_ | key-name | category-name syntax=SAS>

Required Argument
REGISTRY
   stores locale data in the Sasuser library and registers it in the SAS registry. The encoding of the stored locale data is in UNICODE escape. Here is the hierarchy of the saved locale element in the SAS registry: SAS_REGISTRY ⇒ LOCALE ⇒ sas locale ⇒ locale element key locale category ⇒ locale element key.

Optional Arguments
_ALL_
   saves all locale data elements.
key-name
   saves the specified locale element key name and value.

category-name syntax=SAS
   saves all elements in the specified category.

Example: Modifying Locale Data Using the LOCALEDATA Procedure

Example 1: Modifying Locale Data Using the LOCALEDATA Procedure
Features:

PROC LOCALEDATA statement
LOAD statement
MODIFY statement
SAVE statement
CONTENTS statement
DATA step
PUT statement

The following program modifies locale data. SAS is invoked twice.

Modifying Locale Definitions

PROC LOCALEDATA;
LOAD SASLOCALE;
MODIFY KEY=QTR1 VALUE='QTR1';
MODIFY KEY=QTR2 VALUE='QTR2';
MODIFY KEY=QTR3 VALUE='QTR3';
MODIFY KEY=QTR4 VALUE='QTR4';
MODIFY KEY=DATE_SHORT_FORMAT VALUE='%Y//%m//%d';
MODIFY KEY=DATE_YEAR_FORMAT VALUE='Year(%Y)';
SAVE REGISTRY / _ALL_ syntax=SAS;
CONTENTS _ALL_;
QUIT;
data _null_;a = 19208;
put a nldate. / a nldateyq. / a nldateyr.;run;
Program Description

Invoke the LOCALEDATA procedure.

   PROC LOCALEDATA;

Load the locale definitions from the local locale database.

   LOAD SASLOCALE;

Modify the definition values.

   MODIFY KEY=QTR1 VALUE='QTR1';
   MODIFY KEY=QTR2 VALUE='QTR2';
   MODIFY KEY=QTR3 VALUE='QTR3';
   MODIFY KEY=QTR4 VALUE='QTR4';
   MODIFY KEY=DATE_SHORT_FORMAT VALUE='%Y/%m/%d';
   MODIFY KEY=DATE_YEAR_FORMAT VALUE='Year(%Y)';

Save the modified definitions in the registry. This action overwrites the definitions that are in the registry so that there are no duplicate key values.

   SAVE REGISTRY / _ALL_ syntax=SAS;

View the modified locale definitions from the registry.

   CONTENTS _ALL_;

Invoke the QUIT command to end processing.

   QUIT;

Create a DATA step for testing the modified locale definitions.

   data _null_;

Assign a value of 19208 to variable a.

   a = 19208;

Generate the value of variable a using the formats NLDATE., NLDATEYQ., and NLDATEYR.

   put a nldate. / a nldateyq. / a nldateyr.;
   run;
SAS Log

1 PROC LOCALEDATA;
2 /* Load locale definition from locale DB */
3 LOAD SASLOCALE;
4
5 /* Change the value of the elements */
6 MODIFY KEY=QTR1 VALUE='QTR1';
7 MODIFY KEY=QTR2 VALUE='QTR2';
8 MODIFY KEY=QTR3 VALUE='QTR3';
9 MODIFY KEY=QTR4 VALUE='QTR4';
10 MODIFY KEY=DATE_SHORT_FORMAT VALUE='%Y/%m/%d';
11 MODIFY KEY=DATE_YEAR_FORMAT VALUE='Year(%Y)' ;
12 /* Store the locale definition into registry, this will override the existing definition,
13 ! so there is no duplicate key issue */
14 SAVE REGISTRY / _ALL_ syntax=SAS;
15
16 /* View the locale definition from registry */
17 CONTENTS _ALL_
LOCALE="English_UnitedStates"
LANGUAGE="English"
LANGUAGE_SCRIPT="en"
TERRITORY="UnitedStates"
LANGID="en"
LOCID="US"
DATESTYLE="MDY"
PAPERSIZE="LETTER"
FTITLE=""
FTEXT=""
SIMFONT=""
SORTSEQ=""
MESSAGES=""
FORMATNAME_DATE="NLDATE16." 
FORMATNAME_DATETIME="NLDATM24." 
FORMATNAME_TIME="NLTIMES8." 
FORMATNAME_NUMERIC="BEST12." 
FORMATNAME_PERCENT="PERCENT12." 
FONT_SERIF="COMPLEX" 
FONT_SANSSERIF="SIMPLEX" 
FONT_CURSIVE="ITALIC" 
FONT_FANTASY="BRUSH" 
FONT_MONOSPACE="SIMPLEX" 
SIMPLEX="SIMPLEX" 
COMPLEX="COMPLEX" 
SWISS="SWISS" 
ITALIC="ITALIC" 
DATE_FORMAT="%b %d, %Y" 
DATE_SHORT_FORMAT="%Y/%m/%d" 
DATE_MEDIUM_FORMAT="%b %e, %Y" 
DATE_LONG_FORMAT="%b %e, %Y" 
DATE_FULL_FORMAT="%a, %b %e, %Y" 
DATE_FULL_AB_FORMAT="%a, %b %e, %Y" 
DATE_YMM_FORMAT="%b %Y" 
DATE_YM_MONTH_FORMAT="%b %Y" 
DATE_MD_MONTH_FORMAT="%b %d" 
DATE_YEAR_FORMAT="Year(%Y)" 
DATE_YEAR_SHORT_FORMAT="%Y" 
DATE_YQ_YEAR_FORMAT="%q %Y" 
DATE_YQ_YEAR_FORMAT="%q %Y" 
DATE_YW_MONTH_FORMAT="Week %U %Y" 
DATE_YM_MONTH_FORMAT="%Y/%m/%d"
Example 1: Modifying Locale Data Using the LOCALEDATA Procedure

DATETIME_AMPM_FORMAT="%B %e, %Y %I:%M:%S %p"
DATETIME_AMPM_AB_FORMAT="%b %e, %Y %I:%M:%S %p"
DATETIME_FORMAT="%d%b%Y:%H:%M:%S"
DATETIME_SHORT_FORMAT="%o/%e/%Y %I:%M:%S %p"
DATETIME_MEDIUM_FORMAT="%b %e, %Y %I:%M:%S %p"
DATETIME_LONG_FORMAT="%B %e, %Y %I:%M:%S %p"
DATETIME_FULL_FORMAT="%A, %B %e, %Y %I:%M:%S %p"
DATETIME_FULL_AB_FORMAT="%a, %b %e, %Y %I:%M:%S %p"
TIME_AMPM_FORMAT="%I:%M:%S %p"
TIME_FORMAT="%H:%M:%S"
TIME_SHORT_FORMAT="%I:%M %p"
TIME_MEDIUM_FORMAT="%I:%M:%S %p"
TIME_LONG_FORMAT="%I:%M:%S %p %Z"
TIME_FULL_FORMAT="%I:%M:%S %p %Z"
DATETIME_PATTERN="%x %X"
ABMON01="Jan"
ABMON02="Feb"
ABMON03="Mar"
ABMON04="Apr"
ABMON05="May"
ABMON06="Jun"
ABMON07="Jul"
ABMON08="Aug"
ABMON09="Sep"
ABMON10="Oct"
ABMON11="Nov"
ABMON12="Dec"
MON01="January"
MON02="February"
MON03="March"
MON04="April"
MON05="May"
MON06="June"
MON07="July"
MON08="August"
MON09="September"
MON10="October"
MON11="November"
MON12="December"
SA_ABMON01="Jan"
SA_ABMON02="Feb"
SA_ABMON03="Mar"
SA_ABMON04="Apr"
SA_ABMON05="May"
SA_ABMON06="Jun"
SA_ABMON07="Jul"
SA_ABMON08="Aug"
SA_ABMON09="Sep"
SA_ABMON10="Oct"
SA_ABMON11="Nov"
SA_ABMON12="Dec"
SA_MON01="January"
SA_MON02="February"
SA_MON03="March"
SA_MON04="April"
SA_MON05="May"
SA_MON06="June"
SA_MON07="July"
SA_MON08="August"
SA_MON09="September"
SA_MON10="October"
SA_MON11="November"
SA_MON12="December"
ABDAY1 = "Sun"
ABDAY2 = "Mon"
ABDAY3 = "Tue"
ABDAY4 = "Wed"
ABDAY5 = "Thu"
ABDAY6 = "Fri"
ABDAY7 = "Sat"
DAY1 = "Sunday"
DAY2 = "Monday"
DAY3 = "Tuesday"
DAY4 = "Wednesday"
DAY5 = "Thursday"
DAY6 = "Friday"
DAY7 = "Saturday"
SA_ABDAY1 = "Sun"
SA_ABDAY2 = "Mon"
SA_ABDAY3 = "Tue"
SA_ABDAY4 = "Wed"
SA_ABDAY5 = "Thu"
SA_ABDAY6 = "Fri"
SA_ABDAY7 = "Sat"
SA_DAY1 = "Sunday"
SA_DAY2 = "Monday"
SA_DAY3 = "Tuesday"
SA_DAY4 = "Wednesday"
SA_DAY5 = "Thursday"
SA_DAY6 = "Friday"
SA_DAY7 = "Saturday"
ABQTR1 = "Q1"
ABQTR2 = "Q2"
ABQTR3 = "Q3"
ABQTR4 = "Q4"
QTR1 = "QTR1"
QTR2 = "QTR2"
QTR3 = "QTR3"
QTR4 = "QTR4"
SA_ABQTR1 = "Q1"
SA_ABQTR2 = "Q2"
SA_ABQTR3 = "Q3"
SA_ABQTR4 = "Q4"
SA_QTR1 = "1st quarter"
SA_QTR2 = "2nd quarter"
SA_QTR3 = "3rd quarter"
SA_QTR4 = "4th quarter"
AM = "AM"
PM = "PM"
DATE_SEP = "/
FIRST_DAY_OF_WEEK = "0"
INT_CURRENCY_SYMBOL = "USD"
CURRENCY_SYMBOL = "$"
MON_DECIMAL_POINT = "."
MON_THOUSANDS_SEP = ","
MON_GROUPING = "3"
MON_POSITIVE_SIGN = ""
MON_NEGATIVE_SIGN = "-
MON_INT_FRAC_DIGITS = "2"
MON_FRAC_DIGITS = "2"
MON_P_CS_PRECEDES = "1"
MON_P_SEP_BY_SPACE = "0"
MON_N_CS_PRECEDES = "1"
MON_N_SEP_BY_SPACE = "0"
MON_P_SIGN_POSN = "1"
MON_N_SIGN_POSN = "0"
Example 1: Modifying Locale Data Using the LOCALEDATA Procedure

```
CURR_FMT_L="#,##0.00;(#,##0.00)"
CURR_FMT_I="#,##0.##;(#,##0.##)"
NUM_DECIMAL_POINT="."
NUM_THOUSANDS_SEP="";
NUM_GROUPING="3"
NUM_POSITIVE_SIGN="";
NUM_NEGATIVE_SIGN="-"
NUM_P_CS_PRECEDES="1"
NUM_P_SEP_BY_SPACE="0"
NUM_K_CS_PRECEDES="1"
NUM_K_SEP_BY_SPACE="0"
NUM_P_SIGN_POSN="1"
NUM_N_SIGN_POSN="1"
DEC_FMT_L="#,##0.###"
DEC_FMT_I="#,##0.#"
PCT_FMT_L="#,##0%"
PCT_FMT_I="#,##0.#%"
HEIGHT="279"
WIDTH="216"
18
19 QUIT;

NOTE: PROCEDURE LOCALEDATA used (Total process time):
real time 0.24 seconds
cpu time 0.18 seconds

20
21 data _null_;
22 a = 19208 ;
23 put a nldate. / a NLDATEYQ. / a NLDATEYR. ;
24 run ;

August 03, 2012
3rd quarter 2012
2012

NOTE: DATA statement used (Total process time):
real time 2.96 seconds
cpu time 0.28 seconds
```
Overview: TRANTAB Procedure

What Does the TRANTAB Procedure Do?

The TRANTAB procedure creates, edits, and displays customized translation tables. In addition, you can use PROC TRANTAB to view and modify translation tables that are supplied by SAS. These SAS supplied tables are stored in the Sashelp.Host catalog. Any translation table that you create or customize is stored in your Sasuser.Profile catalog. Translation tables have an entry type of TRANTAB.
**Translation tables** are operating environment-specific SAS catalog entries that translate the values of one (coded) character set to another. A translation table has two halves: table 1 provides a translation, such as ASCII to EBCDIC; table 2 provides the inverse (or reverse) translation, such as EBCDIC to ASCII. Each half of a translation table is an array of 256 two-digit **positions**, each of which contains a one-byte unsigned number that corresponds to a coded character.

SAS uses translation tables to
- determine the collating sequence in the **SORT** procedure
- facilitate data communications between the operating environment and a graphics device when you run **SAS/GRAPH** software in an IBM environment
- accommodate national language character sets other than U.S. English.

**PROC TRANTAB** produces no output. It can display translation tables and notes in the SAS log.

---

**Note:** Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the **LOCALE=** system option to improve the direct use of translation tables. SAS 9.2 supports the **TRANTAB** procedure for backward compatibility. However, using the **LOCALE=** system option is preferred in later SAS releases. **PROC TRANTAB** is an interactive procedure. Once you submit a **PROC TRANTAB** statement, you can continue to enter and execute statements without repeating the **PROC TRANTAB** statement. To terminate the procedure, submit a **QUIT** statement or another **DATA** or **PROC** statement.

---

**Concepts: TRANTAB Procedure**

**Understanding Translation Tables and Character Sets for PROC TRANTAB**

The $K^{th}$ element in a translation table corresponds to the $K^{th}$ element of an ordered character set. For example, position 00 (which is byte 1) in a translation table contains a coded value that corresponds to the first element of the ordered character set. To determine the position of a character in your operating environment's character set, use the **RANK** function. The following example shows how to use **RANK**:

```sas
data _null_
  x=rank('a');
  put "The position of a is " x ".";
run;
```

The SAS log prints the following message: *The position of a is 97.*

Each position in a translation table contains a hexadecimal number that is within the range of 0 ("00"x) to 255 ("FF"x). Hexadecimal values always end with an x. You can represent one or more consecutive hexadecimal values within quotation marks followed by a single x. For example, a string of three consecutive hexadecimal
values can be written as '08090A'x. The SAS log displays each row of a translation table as 16 hexadecimal values enclosed in quotation marks followed by an x. The SAS log also lists reference numbers in the vertical and horizontal margins that correspond to the positions in the table. “Example 1: Viewing a Translation Table” on page 793 shows how the SAS log displays a translation table.

Storing Translation Tables with PROC TRANTAB

When you use PROC TRANTAB to create a customized translation table, the procedure automatically stores the table in your Sasuser.Profile catalog. This lets you use customized translation tables without affecting other users. When you specify the translation table in the SORT procedure or in a GOPTIONS statement, the software first looks in your Sasuser.Profile catalog to find the table. If the specified translation table is not in your Sasuser.Profile catalog, the software looks in the Sashelp.Host catalog.

To access your translation table globally, your SAS Installation Representative must copy the table from your Sasuser.Profile catalog (using the CATALOG procedure) to the Sashelp.Host catalog. If the table is not found there, the software continues to search in Sashelp.Locale for it.

Modifying SAS Translation Tables with PROC TRANTAB

If a translation table that is provided by SAS does not meet your needs, you can use PROC TRANTAB to edit it and create a new table. That is, you can issue the PROC TRANTAB statement that specifies the SAS table, edit the table, and then save the table using the SAVE statement. The modified translation table is saved in your Sasuser.Profile catalog. If you are a SAS Installation Representative, you can modify a translation table with PROC TRANTAB and then use the CATALOG procedure to copy the modified table from your Sasuser.Profile catalog to the Sashelp.Host catalog, as shown in the following example:

```
proc catalog c=sasuser.profile;
   copy out=sashelp.host entrytype=trantab;
run;
```

You can use PROC TRANTAB to modify translation tables stored in the Sashelp.Host catalog only if you have update (or write) access to that data library and catalog.

Using Translation Tables Outside PROC TRANTAB

Using Translation Tables in the SORT Procedure

PROC SORT uses translation tables to determine the sort’s collating sequence. You can specify an alternative translation table with the SORTSEQ= option of PROC SORT. For example, if your operating environment sorts with the EBCDIC sequence by default, and you want to sort with the ASCII sequence, you can issue the following statement to specify the ASCII translation table:
proc sort sortseq=ascii;
run;

You can also create a customized translation table with PROC TRANTAB and specify the new table with PROC SORT. This table is useful when you want to specify sorting sequences for languages other than U.S. English.

See “Example 6: Using Different Translation Tables for Sorting” on page 803 for an example that uses translation tables to sort data in different ways. For information about the tables available for sorting and the SORTSEQ= option, see "SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 708.

Transcoding External Files

Translation tables are implicitly set by the LOCALE= system option. They are used only for transcoding external files.

You can list the translation tables with the OPTIONS procedure. The following example uses the z/OS environment with a LOCALE= system option set to en_US:

```
TRANTAB=(eol1wlt1,wlt1eol1,elat_ucs,elat_lcs,elat_ccl,,,elat_scc)
```

Using Translation Tables in SAS/GRAPH Software

In SAS/GRAPH software, translation tables are most commonly used in an IBM operating environment, where tables are necessary because graphics commands must leave IBM operating environments in EBCDIC representation but must reach asynchronous graphics devices in ASCII representation. Specifically, SAS/GRAPH software builds the command stream for these devices internally in ASCII representation. However, it must convert the commands to EBCDIC representation before they can be given to the communications software for transmission to the device. SAS/GRAPH software uses a translation table internally to make the initial conversion from ASCII to EBCDIC. The communications software then translates the command stream back to ASCII representation before it reaches the graphics device.

Translation tables are operating environment-specific. In most cases, you can simply use the default translation table, SASGTAB0, or one of the SAS supplied graphics translation tables. However, if these tables cannot perform all of the translation correctly, you can create your own translation table with PROC TRANTAB. The SASGTAB0 table might fail to do the translation correctly when it encounters characters from languages other than U.S. English.

To specify an alternative translation table for SAS/GRAPH software, you can either use the TRANTAB= option in a GOPTIONS statement or modify the TRANTAB device parameter in the device entry. For example, the following GOPTIONS statement specifies the GTABTCAM graphics translation table:

```
  goptions trantab=gtabtcam;
```

Translation tables used in SAS/GRAPH software perform both device-to-operating environment translation and operating environment-to-device translation. Therefore, a translation table consists of 512 bytes, with the first 256 bytes used to perform device-to-operating environment translation (ASCII to EBCDIC on IBM mainframes) and the second 256 bytes used to perform operating environment-to-device translation (EBCDIC to ASCII on IBM mainframes). For PROC TRANTAB, the area of a translation table for device-to-operating environment translation is considered to
be table 1, and the area for operating environment-to-device translation is considered to be table 2. See “Example 1: Viewing a Translation Table” on page 793 for a listing of the ASCII translation table (a SAS provided translation table), which shows both areas of the table.

For operating environments other than IBM mainframes, translation tables can be used to translate specific characters in the data stream that are created by the driver. For example, if the driver normally generates a vertical bar in the data stream, but you want another character to be generated in place of the vertical bar, you can create a translation table that translates the vertical bar to an alternate character.

SAS/GRAPH software also uses key maps and device maps to map codes generated by the keyboard to specified characters and to map character codes to codes required by the graphics output device. These maps are specific to SAS/GRAPH software. For more information, contact SAS Technical Support.

Syntax: TRANTAB Procedure

Tip: Supports RUN group processing

PROC TRANTAB TABLE=table-name <NLS>;
CLEAR <ONE | TWO | BOTH>;
INVERSE;
LIST <ONE | TWO | BOTH>;
LOAD TABLE=table-name <NLS>;
REPLACE position value-1 < …value-n>;
SAVE <TABLE=table-name> <CAT=libname.catalog> > <ONE | TWO | BOTH>;
SWAP;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC TRANTAB</td>
<td>Creates, edits, or displays a translations table</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Sets all positions in the translation table to 0</td>
</tr>
<tr>
<td>INVERSE</td>
<td>Creates an inverse of table 1</td>
</tr>
<tr>
<td>LIST</td>
<td>Displays a translation table in hexadecimal representation</td>
</tr>
<tr>
<td>LOAD</td>
<td>Loads a translation table into memory for editing</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Replaces the characters in a translation table with specified values</td>
</tr>
<tr>
<td>SAVE</td>
<td>Saves the translation table in your Sasuser.Profile catalog</td>
</tr>
<tr>
<td>SWAP</td>
<td>Exchanges table 1 with table 2</td>
</tr>
</tbody>
</table>
PROC TRANTAB Statement

Creates, edits, or displays a translation table.

Tip: If there is an incorrect table name in the PROC TRANTAB statement, use the LOAD statement to load the correct table. You do not need to reinvoke PROC TRANTAB. New tables are not stored in the catalog until you issue the SAVE statement, so you do not have unwanted tables in your catalog.

Syntax

PROC TRANTAB TABLE=table-name <NLS>;

Required Argument

TABLE=table-name
specifies the translation table to create, edit, or display the table. The specified table name must be a valid one-level SAS name with no more than eight characters.

Optional Argument

NLS
specifies that the table listed in the TABLE= argument is one of five special internal translation tables provided with SAS. You must use the NLS option when you specify one of the five special tables in the TABLE= argument. NLS stands for National Language Support. This option and the associated translation tables provide a method to translate characters that exist in languages other than U.S. English. To use the modified NLS table, specify its name in the SAS system option TRANTAB=. When you load one of these special translation tables, the SAS log displays a note stating that table 2 is uninitialized. That is, table 2 is an empty table that contains all 0s. PROC TRANTAB does not use table 2 for translation in these special cases.

SASXPT
specifies the local-to-transport format translation table (used by the CPORT procedure).

SASLCL
specifies the transport-to-local format translation table (used by the CIMPORT procedure).

SASUCS
specifies the lowercase-to-uppercase translation table (used by the UPCASE function).

SASLCS
specifies the uppercase-to-lowercase translation table (used by the LOWCASE macro).

SASCCL
specifies the character classification table (used internally), which contains flag bytes that correspond to each character position. These positions indicate the class or classes to which each character belongs.
CLEAR Statement
Sets all positions in the translation table to 0; used when you create a new table.

Syntax
CLEAR <ONE | TWO | BOTH>;

Optional Argument
ONE | TWO | BOTH
  ONE
clears table 1.
  TWO
clears table 2.
  BOTH
clears both table 1 and table 2.

Default  ONE

INVERSE Statement
Creates an inverse of table 1 in a translation table. The INVERSE statement creates table 2.

Syntax
INVERSE;

Details
The INVERSE statement does not preserve multiple translations. Suppose table 1 has 2 (or more) different characters translated to the same value; for example, "A" and "B" are both translated to "1". For table 2, the INVERSE statement uses the last translated character for the value. "1" is always translated to "B" and not "A", assuming that "A" appears before "B" in the first table. Sort programs in SAS require an inverse table for proper operation.

LIST Statement
Displays a translation table in hexadecimal representation in the SAS log.
Syntax

```
LIST <ONE | TWO | BOTH>;
```

Optional Argument

```
ONE | TWO | BOTH
```

`ONE`

displays table 1.

`TWO`

displays table 2.

`BOTH`

displays both table 1 and table 2.

Default: **ONE**

---

**LOAD Statement**

Loads a translation table into memory for editing.

**Tips:**

- Use the LOAD statement when you specify an incorrect table name in the PROC TRANTAB statement. You can specify the correct name without reinvoking the procedure.
- Use the LOAD statement to edit multiple translation tables in a single PROC TRANTAB step. (Be sure to save the first table before you load another one.)

Syntax

```
LOAD TABLE=table-name <NLS>;
```

Required Argument

```
TABLE=table-name
```

specifies the name of an existing translation table to edit. The specified table name must be a valid one-level SAS name.

Optional Argument

```
NLS
```

specifies that the table listed in the TABLE= argument is one of five special internal translation tables that are provided with SAS. You must use the NLS option when you specify one of the five special tables in the TABLE= argument:

- **SASXPT**
  
specifies the local-to-transport format translation table.
- **SASLCL**
  
specifies the transport-to-local format translation table.
- **SASUCS**
  
specifies the lowercase-to-uppercase translation table.
SASLCS specifies the uppercase-to-lowercase translation table.

SASCCL specifies the character classification table, which contains flag bytes that correspond to each character position. These positions indicate the class or classes to which each character belongs.

NLS stands for National Language Support. This option and the associated translation tables provide a method to map characters from languages other than English to programs, displays, and files. When you load one of these special translation tables, the SAS log displays a note that states that table 2 is uninitialized. That is, table 2 is an empty table that contains all 0s. PROC TRANTAB does not use table 2 for translation in these special cases.

**REPLACE Statement**

Replaces characters in a translation table with the specified values, starting at the specified position.

Alias: REP

Tip: To save edits, you must issue the SAVE statement.

**Syntax**

```
REPLACE position value-1 < ...value-n>;
```

**Required Arguments**

- `position` specifies the position in a translation table where the replacement begins. The editable positions in a translation table begin at decimal 0 and end at decimal 255. To specify the position, you can do either of the following:
  - Use a decimal or hexadecimal value to specify an actual location. If you specify a decimal value (for example, 20), PROC TRANTAB locates position 20 in the table, which is byte 21. If you specify a hexadecimal value (for example, '14'x), PROC TRANTAB locates the decimal position that is equivalent to the specified hexadecimal value, which in this case is position 20 (or byte 21) in the table.
  - Use a quoted character. PROC TRANTAB locates the quoted character in the table (that is, the quoted character's hexadecimal value) and uses that character's position as the starting position. For example, specifying the following REPLACE statement replaces the first occurrence of the hexadecimal value for "a" and the next two hexadecimal values with the hexadecimal equivalent of "ABC":

    ```
    replace 'a' 'ABC';
    ```

This action is useful when you want to locate alphabetic and numeric characters, but you do not know their actual location. If the quoted character is not found, PROC TRANTAB displays an error message and ignores the statement.

To edit positions 256 through 511 (table 2), follow these steps:

1. Issue the SWAP statement.
2. Issue the appropriate REPLACE statement.
3. Issue the SWAP statement again to reposition the table.

\textit{value-1 < \ldots value-n>}
is one or more decimal, hexadecimal, or character constants that specify the actual value to place in the table, starting at position. You can also use a mixture of these values. That is, you can specify a decimal, a hexadecimal, and a character value in one REPLACE statement. "Example 3: Editing By Specifying a Decimal Value for the Starting Position" on page 797 shows a mixture of all three types of values in the REPLACE statement.

### SAVE Statement

Saves the translation table in your Sasure.Use.Profile catalog.

**Syntax**

```
SAVE <TABLE=table-name> <CAT=libname.catalog> <ONE | TWO | BOTH>;
```

**Optional Arguments**

- **TABLE=table-name**
  - Specifies the name under which the current table is saved. The name must be a valid one-level SAS name.
  - **Default**: If you omit the TABLE= option, the current table is saved under the name you specify in the PROC TRANTAB statement or in the LOAD statement.

- **CAT=libname.catalog**
  - Specifies the name of the output catalog when you are saving a TRANTAB entry.
  - **ONE | TWO | BOTH**
    - **ONE**: Saves table 1.
    - **TWO**: Saves table 2.
    - **BOTH**: Saves both table 1 and table 2.
    - **Default**: BOTH

### SWAP Statement

Exchanges table 1 with table 2 to enable you to edit positions 256 through 511.
Syntax
SWAP;

Examples: TRANTAB Procedure

Example 1: Viewing a Translation Table

Features: LIST statement

This example uses PROC TRANTAB to display the ASCII translation table supplied by SAS. All examples were produced in the Linux environment.

Set the options and specify a translation table.

options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;

Display both halves of the translation table. The LIST BOTH statement displays both the table that provides the translation and the table that provides the inverse translation.

list both;
**Example 2: Creating a Translation Table**

**Features:**
- Procedures features:
  - LIST statement
  - REPLACE statement
  - SAVE statement

This example uses PROC TRANTAB to create a customized translation table. All examples were produced in the Linux environment.

**Set the system options and specify the translation table to edit.**

```plaintext
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=newtable;
```
Replace characters in the translation table starting at a specified position. The REPLACE statement places the values in the table starting at position 0. You can use hexadecimal values of any length in the REPLACE statement. This example uses strings of length 16 to match how translation tables appear in the SAS log.

```plaintext
replace 0
'00010203a309e57ff9ecc40b0c0d0e0f'x
'10111213a5e008e71819c6c51c1d1e1f'x
'c7f6e9e2e40a171beaebe8efee050607'x
'c6e616f4f6f2b045f66dca2b6a7501a'x
'20e1edf3f1f1d1aababfa22e3c282b7c'x
'265facbdca1abb5f5f5f21242a293bac'x
'2d2f5fa6a6a6d2ba6a62c55f3e3f'x
'2d2f5fa6a6a62b2b2a6a62c55f3e3f'x
'2b6162636465666768692d2ba6a62b2b'x
'2d6a6b6c6d6e6f7071722da62d2b2d2d'x
'2d7e737475767778787a2d2b2b2b2b2b'x
'2b2b2b5f5fa65f5f5f5f5f5f5f5f5f5f'x
'7b4142434445464748495f5f5f5f5f5f'x
'7d4a4b4c4d4e4f5051525f5f5f5f5f5f'x
'5c83535455565758595a5f5f755f5f5f'x
'30313233343536373839b75f6eb25f5f'x
;
```

Save the table. The SAVE statement saves the table under the name that is specified in the PROC TRANTAB statement. By default, the table is saved in your Sasuser.Profile catalog.

```plaintext
save;
```

Display both halves of the translation table in the SAS log. The LIST BOTH statement displays both the table that provides the translation and the table that provides the inverse translation.

```plaintext
list both;
```
Create and edit table 2. Table 2 is empty; that is, it consists entirely of 0s. To create table 2, you can use the INVERSE statement. (See: SAS(R) Statements: Reference)

To edit table 2, you can use the SWAP statement with the REPLACE statement. (See: SAS(R) Statements: Reference)

NOTE: Table specified is NEWTABLE.
WARNING: Table NEWTABLE not found! New table is assumed.
NOTE: NEWTABLE table 1 is uninitialized.
NOTE: NEWTABLE table 2 is uninitialized.

NOTE: Saving table NEWTABLE.
NOTE: NEWTABLE table 2 will not be saved because it is uninitialized.

NEWTABLE table 1:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'000010203A309E57FF9EC04B0C0D0E0F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'10111213A5B008B71819C0C51C1D1E1F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'C7CBE2E040A1718AAB885EFBE050607'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'C9B616F6F6F2FB04FDF6DCA2B6A7501A'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'20B11EDF3F1D1AABABFAD22E3C21287'C'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'265FACBDBCA1ABB5F5F21242A2932BAC'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>'2D25F6A6A662B2AB2AE62C255F1E1F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'2B2E12636465666768692D2A6A62E2B'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'2D6A6B6C6D6E6F7071722DA62D2B2D2D'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'2B2B2B5F5F5F5F5F5F5F5F5F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'7B41243445464748494F5F5F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'7D4A4B4C4D4E4F5F5F5F5F5F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'5C835355555555555555555555555555'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'30313233343536373839375F6E25F5F5'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: NEWTABLE table 2 is uninitialized.

NEWTABLE table 2:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 3: Editing By Specifying a Decimal Value for the Starting Position

Features:
- LIST statement
- REPLACE statement
- SAVE statement

This example edits the translation table that was created in "Example 2: Creating a Translation Table" on page 794. The decimal value specified in the REPLACE statement marks the starting position for the table changes. The vertical arrow in both SAS logs marks the point at which the changes begin. All examples were produced in the Linux environment.

Program 1: Display the Original Table

```sas
options nodate pageno=1 linesize=80 pagesize=60; proc trantab table=newtable;
list one;
```

Program Description

Set the system options and specify the translation table to edit.

```sas
options nodate pageno=1 linesize=80 pagesize=60; proc trantab table=newtable;
```

Display the original table. This LIST statement displays the original NEWTABLE translation table.

```sas
list one;
```

SAS Log

Original NEWTABLE Translation Table
Table specified is NEWTABLE.
NOTE: NEWTABLE table 2 is uninitialized.

NEWTABLE table 1:

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 00 | '00010203A309E57FF9EC60B5C0DD0E0F'x |
| 10 | '101112135E08E71819C6C51C1D1E1F'x |
| 20 | 'C7FCE892E40A171B8BB88E8FE8050B07'x |
| 30 | 'C9E616F4F6F2FB0F04FFD6DC2A2B6A7501A'x |
| 40 | '20E1EDF3FA1DB1AAABFA82B3C282B70C'x |
| 50 | '265FACBD0ACABBB5F5F212D2A293BAC'x |
| 60 | '2D2F5FA6A6A62B22A4C2C55F1E3F'x |
| 70 | 'A62B22B22B22D03A2340273D22'x |
| 80 | '2B6162636465666768692D2B36A62B'x |
| 90 | '2D6A66C6D6E6F7071722DA62D2B2D2D'x |
| B0 | '2B2B25F5FA655F5F5F5F5F5F5F5F'x |
| C0 | '7B4A24234444644646748945F5F5F5F5F'x |
| D0 | '7D4A4B4C4D4R4F5051525F5F15F5F'x |
| E0 | '5C8353545556558595A5F5F5F5F5F5F0'x |
| F0 | '303112133343536373839B75F5F5F5F5F'x |

Program 2: Edit the Table

```
replace 10
  20 10 200 'x' 'ux' '092040'x;

save;

list one;
```

Program Description

Replace characters in the translation table, starting at a specified position. The REPLACE statement starts at position decimal 10, which is byte 11 in the original table, and performs a byte-to-byte replacement with the given values.

```
replace 10
  20 10 200 'x' 'ux' '092040'x;
```

Save the changes. The SAVE statement saves the changes that you made to the NEWTABLE translation table.

```
save;
```

Display the new table. The second LIST statement displays the edited NEWTABLE translation table.

```
list one;
```
SAS Log

Saving table NEWTABLE.
NOTE: NEWTABLE table 2 will not be saved because it is uninitialized.
NEWTABLE table 1:

0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '0010203A09E57FF9C140AC8787578'x
10 '0920401A5008B71819C6C51C1D181'x
20 '17C78E92B40A171BBABEBE8EFEE050607'x
30 'C9E616F46F2F04FD6DCA2B6A7503A'x
40 '20E1E1F3F4D1AABAB8F42E3C82B7C'x
50 '265FA1B2C1A1BBB5F5F2124A293BAC'x
60 '2D2F5A6A6A62A62B2A62C255F183F'x
80 '2B61626E666E6768692D2C2A262B2B'x
90 '2D6A6B6C666F7071722DA62D2B2D2D'x
B0 '2B2B2B5F5F5FA5F5F5F5F5F5F5F5F'x
C0 '7B814243444546474849555F5F5F5F5F'x
D0 '7D4A4B4C4D4F4F5051525F5F5F5F5F'x
E0 '5C8353545555565758595A5F5F5F5F5F'x
F0 '30313233343536373839B75F6E25F5F'x

Output Details

At position 10 (which is byte 11), a vertical arrow denotes the starting point for the changes to the translation table.

At byte 11, decimal 20 (which is hexadecimal 14) replaces hexadecimal C4.
At byte 12, decimal 10 (which is hexadecimal 0A) replaces hexadecimal 0B.
At byte 13, decimal 200 (which is hexadecimal C8) replaces hexadecimal 0C.
At byte 14, character 'x' (which is hexadecimal 78) replaces hexadecimal 0D.
At bytes 15 and 16, characters 'ux' (which are hexadecimal 75 and 78, respectively) replace hexadecimal 0E and 0F.
At bytes 17, 18, and 19, hexadecimal 092040 replaces hexadecimal 101112.

Example 4: Editing By Using a Quoted Character for the Starting Position

Features:
LIST statement
LOAD statement
REPLACE statement
SAVE statement

This example creates a new translation table by editing the ASCII translation table. The first occurrence of the hexadecimal equivalent of the quoted character that was specified in the REPLACE statement is the starting position for the changes to the table. This method differs from “Example 3: Editing By Specifying a Decimal Value for the Starting Position” on page 797 because you do not need to know the exact
position at which to start the changes to the table. PROC TRANTAB finds the correct position for you.

The edited table is saved under a new name. Horizontal arrows in both SAS logs denote the edited rows in the translation table. All examples were produced in the Linux environment.

Program 1: Display the Original Table

```sas
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;
list one;
```

Program Description

Set the system options and specify which translation table to edit.

```sas
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;
```

Display the translation table. The LIST statement displays the original translation table in the SAS log.

```sas
list one;
```

SAS Log

NOTE: Table specified is ASCII.
ASCII table 1:

```
0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '000102030405060708090A0B0C0D0E0F'x
10 '101112131415161718191A1B1C1D1E1F'x
20 '202122232425262728292A2B2C2D2E2F'x
30 '303132333435363738393A3B3C3D3E3F'x
40 '404142434445464748494A4B4C4D4E4F'x
50 '505152535455565758595A5B5C5D5E5F'x
60 '606162636465666768696A6B6C6D6E6F'x
70 '707172737475767778797A7B7C7D7E7F'x
80 '808182838485868788898A8B8C8D8E8F'x
90 '909192939495969798999A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCCDDEDFF'x
C0 'CC1CC2CC3CC4CC5CC6CC7CC8CC9CACCDCDCECF'x
D0 'DD1DD2DD3DD4DD5DD6DD7DD8DD9DADBDCDDEDFF'x
E0 'EOE1EOE2EOE3EOE4EOE5EOE6EOE7EOE8EOE9EAEBECEDEEF'x
F0 'FF0F1F2F3F4F5F6F7F8F9F9FAFBCFDFEFFFF'x
```

Program 2: Edit the Table

```sas
replace 'a' 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
save table=upper;
load table=upper;
list one;
```
Example 5: Creating the Inverse of a Table

Program Description

Replace characters in the translation table, starting at a specified position. The REPLACE statement finds the first occurrence of the hexadecimal "a" (which is 61) and replaces it and the next 25 hexadecimal values with the hexadecimal values for uppercase "A" through "Z."

```
replace 'a' 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

Save your changes. The SAVE statement saves the changes made to the ASCII translation table under the new table name UPPER. The stored contents of the ASCII translation table remain unchanged.

```
save table=upper;
```

Load and display the translation table. The LOAD statement loads the edited translation table UPPER. The LIST statement displays the translation table UPPER in the SAS log.

```
load table=upper;
list one;
```

SAS Log

```
NOTE: Table UPPER being loaded.
UPPER table 1:
  0 1 2 3 4 5 6 7 8 9 A B C D E F
  00 '000102030405060708090A0B0C0D0E0F'x
  10 '101112131415161718191A1B1C1D1E1F'x
  20 '202122232425262728292A2B2C2D2E2F'x
  30 '303132333435363738393A3B3C3D3E3F'x
  40 '404142434445464748494A4B4C4D4E4F'x
  50 '505152535455565758595A5B5C5D5E5F'x
  60 '606162636465666768696A6B6C6D6E6F'x  ←
  70 '707172737475767778797A7B7C7D7E7F'x  ←
  80 '808182838485868788898A8B8C8D8E8F'x
  90 '909192939495969798999A9B9C9D9E9F'x
  A0 'A0A1A2A3A4A5A6A7A8A9AAABACAABAP'x
  B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBCBEBF'x
  C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCDECF'x
  D0 'D0D1D2D3D4D5D6D7D8D9DADBDCEDEEDF'x
  E0 'E0E1E2E3E4E5E6E7E8E9EAEABECEDEEFEFP'x
  F0 'F0F1F2F3F4F5F6F7F8F9FABFCFDCFDFEF'x
```

Example 5: Creating the Inverse of a Table

Features:

- INVERSE statement
- LIST statement
- SAVE statement

This example creates the inverse of the translation table that was created in “Example 4: Editing By Using a Quoted Character for the Starting Position” on page 799. The new translation table that is created in this example is the operating environment-to-device translation for use in data communications.

```
options nodate pageno=1 linesize=80 pagesize=60;
```
Create the inverse translation table, save the tables, and then display them.
The INVERSE statement creates table 2 by inverting the original table 1 (called UPPER). The SAVE statement saves the translation tables. The LIST BOTH statement displays both the original translation table and its inverse.

inverse;
save;
list both;

SAS Log

The INVERSE statement lists in the SAS log all of the multiple translations that it encounters as it inverts the translation table. In "Example 4: Editing By Using a Quoted Character for the Starting Position" on page 799, all the lowercase letters are converted to uppercase in the translation table UPPER, which means that there are two sets of uppercase letters in UPPER. When INVERSE cannot make a translation, PROC TRANTAB fills the value with 00. Note that the inverse of the translation table UPPER has numerous 00 values.

The SAS log lists all the duplicate values that it encounters as it creates the inverse of table 1. To conserve space, most of these messages are deleted in this example.
Example 6: Using Different Translation Tables for Sorting

Features:

PROC SORT statement option: SORTSEQ=

PRINT procedure

This example shows how to specify a different translation table to sort data in an order that is different from the default sort order. Characters that are written in a language other than U.S. English might require a sort order that is different from the default order.
You can use the TRABASE program in the SAS Sample Library to create translation tables for several languages. All examples were produced in the Linux environment.

**Set the SAS system options.**

```sas
options nodate pageno=1 linesize=80 pagesize=60;
```

**Create the TESTSORT data set.** The DATA step creates a SAS data set with four pairs of words, each pair different only in the case of the first letter.

```sas
data testsort;
  input Values $10.;
datalines;
Always
always
Forever
forever
Later
later
Yesterday
yesterday
;
```

**Sort the data in an order that is different from the default sort order.** PROC SORT sorts the data by using the default translation table, which sorts all lowercase words first, then all uppercase words.

```sas
proc sort;
  by values;
run;
```

**Print the data set.** PROC PRINT prints the sorted data set.

```sas
proc print noobs;
  title 'Default Sort Sequence';
run;
```

**SAS Output**

The following output is the output from sorting values with the default translation table. The default sort sequence sorts all the capitalized words in alphabetical order before it sorts any lowercase words.

```
<table>
<thead>
<tr>
<th>Default Sort Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>Values</td>
</tr>
<tr>
<td>Always</td>
</tr>
<tr>
<td>Forever</td>
</tr>
<tr>
<td>Later</td>
</tr>
<tr>
<td>Yesterday</td>
</tr>
<tr>
<td>always</td>
</tr>
<tr>
<td>forever</td>
</tr>
<tr>
<td>later</td>
</tr>
<tr>
<td>yesterday</td>
</tr>
</tbody>
</table>
```

Sort the data according to the translation table UPPER and print the new data set. The SORTSEQ= system option specifies that PROC SORT sort the data according to the customized translation table UPPER, which treats lowercase
and uppercase letters alike. This method is useful for sorting without regard for case. PROC PRINT prints the sorted data set.

```sas
proc sort sortseq=upper;
  by values;
run;
proc print noobs;
  title 'Customized Sort Sequence';
run;
```

**SAS Output**

The following output is the result from sorting values with a customized translation table. The customized sort sequence sorts all the words in alphabetical order, without regard for the case of the first letters.

<table>
<thead>
<tr>
<th>Customized Sort Sequence</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td></td>
</tr>
<tr>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>always</td>
<td></td>
</tr>
<tr>
<td>Forever</td>
<td></td>
</tr>
<tr>
<td>forever</td>
<td></td>
</tr>
<tr>
<td>Later</td>
<td></td>
</tr>
<tr>
<td>later</td>
<td></td>
</tr>
<tr>
<td>Yesterday</td>
<td></td>
</tr>
<tr>
<td>yesterday</td>
<td></td>
</tr>
</tbody>
</table>

**Example 7: Editing Table 1 and Table 2**

**Features:**
- LIST statement
- REPLACE statement
- SAVE statement
- SWAP statement

This example shows how to edit both areas of a translation table. Arrows in the SAS logs mark the rows and columns that are changed. To edit positions 256 through 511 (table 2), you must

- Issue the SWAP statement to have table 2 change places with table 1.
- Issue an appropriate REPLACE statement to make changes to table 2.
- Issue the SWAP statement again to reposition the table.

**Set the SAS system options and specify the translation table.**

```sas
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=upper;
```

**Display the original translation table. The LIST statement displays the original UPPER translation table.**

```sas
list both;
```
Replace characters in the translation table starting at a specified position. The REPLACE statement starts at position 1 and replaces the current value of 01 with '0A':

```
replace 1 '0A'x;
```

Prepare table 2 for editing. The first SWAP statement positions table 2 so that it can be edited. The second REPLACE statement makes the same change in table 2 that was made in table 1.

```
swap;
replace 1 '0A'x;
```

Save and display the tables in their original positions. The second SWAP statement restores table 1 and table 2 to their original positions. The SAVE statement saves both areas of the translation table by default. The LIST statement displays both areas of the table.
swap;
save;
list both;

SAS Log

The upper table 1 is modified. The hexadecimal value '0A' replaces hexadecimal value 01 in byte 2 for both areas of the translation table. Arrows mark the rows and columns of the table in which this change is made.

NOTE: Table specified is UPPER.
UPPER table 1:

0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '000a02030405060708090a0b0c0d0e0f'x <--
10 '101112131415161718191a1b1c1d1e1f'x
20 '202122232425262728292a2b2c2d2e2f'x
30 '303132333435363738393a3b3c3d3e3f'x
40 '404142434445464748494a4b4c4d4e4f'x
50 '505152535455565758595a5b5c5d5e5f'x
60 '606162636465666768696a6b6c6d6e6f'x
70 '707172737475767778797a7b7c7d7e7f'x
80 '808182838485868788898a8b8c8d8e8f'x
90 '909192939495969798999a9b9c9d9e9f'x
A0 'a0a1a2a3a4a5a6a7a8a9aaabacadaeaf'x
B0 'b0b1b2b3b4b5b6b7b8b9babbbcbbdbbebf'x
C0 'c0c1c2c3c4c5c6c7c8c9caccdccccdfef'x
D0 'd0d1d2d3d4d5d6d7d8d9daadbcdcdedeef'x
E0 'e0e1e2e3e4e5e6e7e8e9eaebecedceedeeef'x
F0 'f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff'x

UPPER table 2:

0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '000a02030405060708090a0b0c0d0e0f'x <--
10 '101112131415161718191a1b1c1d1e1f'x
20 '202122232425262728292a2b2c2d2e2f'x
30 '303132333435363738393a3b3c3d3e3f'x
40 '404142434445464748494a4b4c4d4e4f'x
50 '505152535455565758595a5b5c5d5e5f'x
60 '600000000000000000007b7c7d7e7f'x
70 '00000000000000000000000000000000000'x
80 '808182838485868788898a8b8c8d8e8f'x
90 '909192939495969798999a9b9c9d9e9f'x
A0 'a0a1a2a3a4a5a6a7a8a9aaabacadaeaf'x
B0 'b0b1b2b3b4b5b6b7b8b9babbbcbbdbbebf'x
C0 'c0c1c2c3c4c5c6c7c8c9caccdccccdfef'x
D0 'd0d1d2d3d4d5d6d7d8d9daadbcdcdedeef'x
E0 'e0e1e2e3e4e5e6e7e8e9eaebecedceedeeef'x
F0 'f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff'x

Example 7: Editing Table 1 and Table 2
Chapter 21
Values for the LOCALE= System Option ............................................. 811

Chapter 22
SAS System Options for Processing DBCS Data ............................... 827

Chapter 23
Encoding Values in SAS Language Elements ................................... 829

Chapter 24
Encoding Values for a SAS Session ....................................................... 841
Values for the LOCALE= System Option

The following table lists the valid LOCALE= values, specified by using the SAS name or the POSIX name. The alias name is also listed. Some locales do not have an alias.

The Locale to Encoding Mapping Table maps the locale to the default encoding used on various hosts.

### Table 21.1 Values for the LOCALE= System Option

<table>
<thead>
<tr>
<th>SAS Name</th>
<th>POSIX Locale</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans_SouthAfrica</td>
<td>af_ZA</td>
<td>Afrikaans</td>
</tr>
<tr>
<td></td>
<td></td>
<td>af</td>
</tr>
<tr>
<td>Albanian_Alabania</td>
<td>sq_AL</td>
<td>Albanian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sq</td>
</tr>
<tr>
<td>Arabic_Algeria</td>
<td>ar_DZ</td>
<td></td>
</tr>
<tr>
<td>Arabic_Bahrain</td>
<td>ar_BH</td>
<td></td>
</tr>
<tr>
<td>Arabic_Egypt</td>
<td>ar_EG</td>
<td></td>
</tr>
<tr>
<td>Arabic_India</td>
<td>ar_IN</td>
<td></td>
</tr>
<tr>
<td>Arabic_Iraq</td>
<td>ar_IQ</td>
<td></td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>Arabic_Jordan</td>
<td>ar_JO</td>
<td></td>
</tr>
<tr>
<td>Arabic_Kuwait</td>
<td>ar_KW</td>
<td></td>
</tr>
<tr>
<td>Arabic_Lebanon</td>
<td>ar_LB</td>
<td></td>
</tr>
<tr>
<td>Arabic_Libya</td>
<td>ar_LY</td>
<td></td>
</tr>
<tr>
<td>Arabic_Morocco</td>
<td>ar_MA</td>
<td></td>
</tr>
<tr>
<td>Arabic_Oman</td>
<td>ar_OM</td>
<td></td>
</tr>
<tr>
<td>Arabic_Qatar</td>
<td>ar_QA</td>
<td></td>
</tr>
<tr>
<td>Arabic_SaudiArabia</td>
<td>ar_SA</td>
<td></td>
</tr>
<tr>
<td>Arabic_Sudan</td>
<td>ar_SD</td>
<td></td>
</tr>
<tr>
<td>Arabic_Syria</td>
<td>ar_SY</td>
<td></td>
</tr>
<tr>
<td>Arabic_Tunisia</td>
<td>ar_TN</td>
<td></td>
</tr>
<tr>
<td>Arabic_UnitedArabEmirates</td>
<td>ar_AE</td>
<td>Arabic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ar</td>
</tr>
<tr>
<td>Arabic_Yemen</td>
<td>ar_YE</td>
<td></td>
</tr>
<tr>
<td>Basque_Spain</td>
<td>eu_ES</td>
<td>Basque</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Euskara</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eu</td>
</tr>
<tr>
<td>Bengali_India</td>
<td>bn_IN</td>
<td>Bengali</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bn</td>
</tr>
<tr>
<td>Bosnian_BosniaHerzegovina</td>
<td>bs_BA</td>
<td>Bosnian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bs</td>
</tr>
<tr>
<td>Bulgarian_Bulgaria</td>
<td>bg_BG</td>
<td>Bulgarian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bg</td>
</tr>
<tr>
<td>Byelorussian_Belarus</td>
<td>be_BY</td>
<td>Byelorussian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Belarusian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Byelorussian_Belarus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>be</td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Catalan_Spain</td>
<td>ca_ES</td>
<td>Catalan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ca</td>
</tr>
<tr>
<td>Chinese_China</td>
<td>zh_CN</td>
<td>Chinese</td>
</tr>
<tr>
<td></td>
<td></td>
<td>zh_HANS_CN</td>
</tr>
<tr>
<td>Chinese_China</td>
<td>zh_CN</td>
<td>zh_HANT_HK</td>
</tr>
<tr>
<td>Chinese_HongKong</td>
<td>zh_HK</td>
<td>zh_HANT_MO</td>
</tr>
<tr>
<td>Chinese_Macau</td>
<td>zh_MO</td>
<td>zh_HANT_MO</td>
</tr>
<tr>
<td>Chinese_Singapore</td>
<td>zh_SG</td>
<td>zh_HANS_SG</td>
</tr>
<tr>
<td>Chinese_Taiwan</td>
<td>zh_TW</td>
<td>zh_HANT_TW</td>
</tr>
<tr>
<td>Cornish_UnitedKingdom</td>
<td>kw_GB</td>
<td>Cornish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kw</td>
</tr>
<tr>
<td>Croatian_BosniaHerzegovina</td>
<td>hr_BA</td>
<td>Croatian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hr</td>
</tr>
<tr>
<td>Croatian_Croatia</td>
<td>hr_HR</td>
<td>Croatian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hr</td>
</tr>
<tr>
<td>Czech_CzechRepublic</td>
<td>cs_CZ</td>
<td>Czech</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cs</td>
</tr>
<tr>
<td>Danish_Denmark</td>
<td>da_DK</td>
<td>Danish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>da</td>
</tr>
<tr>
<td>Dutch_Belgium</td>
<td>nl_BE</td>
<td>Dutch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>Dutch_Netherlands</td>
<td>nl_NL</td>
<td>Dutch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>English_Australia</td>
<td>en_AU</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>English_Belgium</td>
<td>en_BE</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>English_Botswana</td>
<td>en_BW</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>English_Canada</td>
<td>en_CA</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>English_Caribbean</td>
<td>en_CB</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>English_HongKong</td>
<td>en_HK</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>English_India</td>
<td>en_IN</td>
<td></td>
</tr>
<tr>
<td>English_Ireland</td>
<td>en_IE</td>
<td></td>
</tr>
<tr>
<td>English_Jamaica</td>
<td>en_JM</td>
<td></td>
</tr>
<tr>
<td>English_Malta</td>
<td>en_MT</td>
<td></td>
</tr>
<tr>
<td>English_NewZealand</td>
<td>en_NZ</td>
<td></td>
</tr>
<tr>
<td>English_Philippines</td>
<td>en_PH</td>
<td></td>
</tr>
<tr>
<td>English_Singapore</td>
<td>en_SG</td>
<td>English en</td>
</tr>
<tr>
<td>English_SouthAfrica</td>
<td>en_ZA</td>
<td></td>
</tr>
<tr>
<td>English_UnitedKingdom</td>
<td>en_GB</td>
<td></td>
</tr>
<tr>
<td>English_UnitedStates</td>
<td>en_US</td>
<td>English en</td>
</tr>
<tr>
<td>English_Zimbabwe</td>
<td>en_ZW</td>
<td></td>
</tr>
<tr>
<td>Estonian_Estonia</td>
<td>et_EE</td>
<td>Estonian et</td>
</tr>
<tr>
<td>Faroese_FaroeIslands</td>
<td>fo_FO</td>
<td>Faroese fo</td>
</tr>
<tr>
<td>Finnish_Finland</td>
<td>fi_FI</td>
<td>Finnish fi</td>
</tr>
<tr>
<td>French_Belgium</td>
<td>fr_BE</td>
<td></td>
</tr>
<tr>
<td>French_Canada</td>
<td>fr_CA</td>
<td></td>
</tr>
<tr>
<td>French_France</td>
<td>fr_FR</td>
<td>French fr</td>
</tr>
<tr>
<td>French_Luxembourg</td>
<td>fr_LU</td>
<td></td>
</tr>
<tr>
<td>French_Switzerland</td>
<td>fr_CH</td>
<td></td>
</tr>
<tr>
<td>German_Austria</td>
<td>de_AT</td>
<td></td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>German_Germany</td>
<td>de_DE</td>
<td>German</td>
</tr>
<tr>
<td>German_Liechtenstein</td>
<td>de_LI</td>
<td></td>
</tr>
<tr>
<td>German_Luxembourg</td>
<td>de_LU</td>
<td></td>
</tr>
<tr>
<td>German_Switzerland</td>
<td>de_CH</td>
<td></td>
</tr>
<tr>
<td>Greek_Cyprus</td>
<td>el_CY</td>
<td>Greek</td>
</tr>
<tr>
<td>Greek_Greece</td>
<td>el_GR</td>
<td>el</td>
</tr>
<tr>
<td>Greenlandic_Greenland</td>
<td>kl_GL</td>
<td>Greenlandic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kl</td>
</tr>
<tr>
<td>Hebrew_Israel</td>
<td>he_IL</td>
<td>Hebrew</td>
</tr>
<tr>
<td></td>
<td></td>
<td>he</td>
</tr>
<tr>
<td>Hindi_India</td>
<td>hi_IN</td>
<td>Hindi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hi</td>
</tr>
<tr>
<td>Hungarian_Hungary</td>
<td>hu_HU</td>
<td>Hungarian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hu</td>
</tr>
<tr>
<td>Icelandic_Iceland</td>
<td>is_IS</td>
<td>Icelandic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is</td>
</tr>
<tr>
<td>Indonesian_Indonesia</td>
<td>id_ID</td>
<td>Indonesian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>id</td>
</tr>
<tr>
<td>Irish_Ireland</td>
<td>ga_IE</td>
<td>Irish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ga</td>
</tr>
<tr>
<td>Italian_Italy</td>
<td>it_IT</td>
<td>Italian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>it</td>
</tr>
<tr>
<td>Italian_Switzerland</td>
<td>it_CH</td>
<td></td>
</tr>
<tr>
<td>Japanese_Japan</td>
<td>ja_JP</td>
<td>Japanese</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ja</td>
</tr>
<tr>
<td>Kazakh_Kazakhstan</td>
<td>kk_KZ</td>
<td>Kazakh</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kk</td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Korean_Korea</td>
<td>ko_KR</td>
<td>Korean, ko, Korean_SouthKorea</td>
</tr>
<tr>
<td>Latvian_Latvia</td>
<td>lv_LV</td>
<td>Latvian, lv</td>
</tr>
<tr>
<td>Lithuanian_Lithuania</td>
<td>lt_LT</td>
<td>Lithuanian, lt</td>
</tr>
<tr>
<td>Macedonian_Macedonia</td>
<td>mk_MK</td>
<td>Macedonian, mk</td>
</tr>
<tr>
<td>Malay_Malaysia</td>
<td>ms_MY</td>
<td>Malay, ms</td>
</tr>
<tr>
<td>Maltese_Malta</td>
<td>mt_MT</td>
<td>Maltese, mt</td>
</tr>
<tr>
<td>Marathi_India</td>
<td>mr_IN</td>
<td>Marathi, mr</td>
</tr>
<tr>
<td>NorwegianBokmal_Norway</td>
<td>nb_NO</td>
<td>NorwegianBokmal, nb</td>
</tr>
<tr>
<td>NorwegianNynorsk_Norway</td>
<td>nn_NO</td>
<td>NorwegianNynorsk, nn</td>
</tr>
<tr>
<td>Norwegian_Norway</td>
<td>no_NO</td>
<td>Norwegian, no</td>
</tr>
<tr>
<td>Persian_India</td>
<td>fa_IN</td>
<td>Persian, fa</td>
</tr>
<tr>
<td>Persian_Iran</td>
<td>fa_IR</td>
<td>Persian, fa</td>
</tr>
<tr>
<td>Polish_Poland</td>
<td>pl_PL</td>
<td>Polish, pl</td>
</tr>
<tr>
<td>Portuguese_Brazil</td>
<td>pt_BR</td>
<td>Portuguese</td>
</tr>
<tr>
<td>Portuguese_Portugal</td>
<td>pt_PT</td>
<td>Portuguese</td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Romanian_Romania</td>
<td>ro_RO</td>
<td>Romanian</td>
</tr>
<tr>
<td>Russian_Russia</td>
<td>ru_RU</td>
<td>Russian</td>
</tr>
<tr>
<td>Russian_Ukraine</td>
<td>ru_UA</td>
<td>ru</td>
</tr>
<tr>
<td>Serbian_BosniaHerzegovina</td>
<td>sr_BA</td>
<td></td>
</tr>
<tr>
<td>Serbian_Montenegro</td>
<td>sr_ME</td>
<td></td>
</tr>
<tr>
<td>Serbian_Serbia</td>
<td>sr_RS</td>
<td>sr_RS</td>
</tr>
<tr>
<td>SerbianLatin_BosniaHerzegovina</td>
<td>sh_BA</td>
<td>sh_RS</td>
</tr>
<tr>
<td>SerbianLatin_Montenegro</td>
<td>sh_ME</td>
<td>sh_LATN</td>
</tr>
<tr>
<td>Slovak_Slovakia</td>
<td>sk_SK</td>
<td>Slovak</td>
</tr>
<tr>
<td>Slovenian_Slovenia</td>
<td>sl_SI</td>
<td>Slovenian</td>
</tr>
<tr>
<td>Spanish_Argentina</td>
<td>es_AR</td>
<td></td>
</tr>
<tr>
<td>Spanish_Bolivia</td>
<td>es_BO</td>
<td></td>
</tr>
<tr>
<td>Spanish_Chile</td>
<td>es_CL</td>
<td></td>
</tr>
<tr>
<td>Spanish_Colombia</td>
<td>es_CO</td>
<td></td>
</tr>
<tr>
<td>Spanish_CostaRica</td>
<td>es_CR</td>
<td></td>
</tr>
<tr>
<td>Spanish_DominicanRepublic</td>
<td>es_DO</td>
<td></td>
</tr>
<tr>
<td>Spanish_Ecuador</td>
<td>es_EC</td>
<td></td>
</tr>
<tr>
<td>Spanish_ElSalvador</td>
<td>es_SV</td>
<td></td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Spanish_Guatemala</td>
<td>es_GT</td>
<td></td>
</tr>
<tr>
<td>Spanish_Honduras</td>
<td>es_HN</td>
<td></td>
</tr>
<tr>
<td>Spanish_Mexico</td>
<td>es_MX</td>
<td></td>
</tr>
<tr>
<td>Spanish_Nicaragua</td>
<td>es_NI</td>
<td></td>
</tr>
<tr>
<td>Spanish_Panama</td>
<td>es_PA</td>
<td></td>
</tr>
<tr>
<td>Spanish_Paraguay</td>
<td>es_PY</td>
<td></td>
</tr>
<tr>
<td>Spanish_Peru</td>
<td>es_PE</td>
<td></td>
</tr>
<tr>
<td>Spanish_PuertoRico</td>
<td>es_PR</td>
<td></td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>es_ES</td>
<td>Spanish es</td>
</tr>
<tr>
<td>Spanish_UnitedStates</td>
<td>es_US</td>
<td></td>
</tr>
<tr>
<td>Spanish_Uruguay</td>
<td>es_UY</td>
<td></td>
</tr>
<tr>
<td>Spanish_Venezuela</td>
<td>es_VE</td>
<td></td>
</tr>
<tr>
<td>Swedish_Sweden</td>
<td>sv_SE</td>
<td>Swedish sv</td>
</tr>
<tr>
<td>Tagalog_Philippines</td>
<td>tl_PH</td>
<td>Tagalog Filipino Filipino_Philippines tl</td>
</tr>
<tr>
<td>Tamil_India</td>
<td>ta_IN</td>
<td>Tamil ta</td>
</tr>
<tr>
<td>Telugu_India</td>
<td>te_IN</td>
<td>Telugu te</td>
</tr>
<tr>
<td>Thai_Thailand</td>
<td>th_TH</td>
<td>Thai th</td>
</tr>
<tr>
<td>Turkish_Turkey</td>
<td>tr_TR</td>
<td>Turkish tr</td>
</tr>
</tbody>
</table>
## Locale-to-Encoding Mapping

This table lists the valid POSIX values, and maps the locale to the default encoding used on various hosts. The settings for DFLANG, DATESTYLE, and PAPERSIZE system options are set automatically.

Here is an example:

```
sas -locale arabic_algeria
```

When the Arabic_Algeria LOCALE= value is specified, corresponding default settings for the system options are as follows:

- DFLANG=English
- DATESTYLE=DMY
- PAPERSIZE=A4

**Table 21.2** Default Values for DFLANG, DATESTYLE, and PAPERSIZE System Options Based on the LOCALE= System Option

<table>
<thead>
<tr>
<th>POSIX Locale</th>
<th>Windows Encoding</th>
<th>UNIX Encoding</th>
<th>z/OS Encoding</th>
<th>DFLANG</th>
<th>DATESTYLE</th>
<th>PAPERSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>af_ZA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ar_AE</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_BH</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_DZ</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_EG</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IQ</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_JO</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_KW</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG =</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>----------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ar_LB</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_LY</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_MA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_OM</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_QA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SD</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SY</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_TN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_YE</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>be_BY</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-102</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>bg_BG</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-102</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>bn_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>bs_BA</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ca_ES</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-114</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>cs_CZ</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Czech</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>da_DK</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114</td>
<td>Danish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_AT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114</td>
<td>Swiss_German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_DE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_LI</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG</td>
<td>DATESTYLE</td>
<td>PAPERSIZE</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>--------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>de_LU</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 1</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>el_CY</td>
<td>Wgreek</td>
<td>greek</td>
<td>Open_ed-875</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>el_GR</td>
<td>wgreek</td>
<td>greek</td>
<td>open_ed-875</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_AU</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 8</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BW</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_CA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>en_CB</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>en_GB</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_HK</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IN</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_JM</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_MT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>Open_ed-114 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_NZ</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_PH</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>en_SG</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_US</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG =</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>en_ZA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_ZW</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_AR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_BO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_CL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_DO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_EC</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_ES</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 5</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_GT</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_HN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>es_MX</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_NI</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_PR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG =</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>es_PY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_SV</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_US</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_UY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_VE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>et_EE</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-1122</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>eu_ES</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1145</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-1097</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IR</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-1097</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fi_FI</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1143</td>
<td>Finnish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fo_FO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_BE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_CA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>French</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>fr_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>Swiss_French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_FR</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1147</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr-LU</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1147</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ga_IE</td>
<td>Wlatin1</td>
<td>Latin8</td>
<td>Open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>he_IL</td>
<td>whebrew</td>
<td>hebrew</td>
<td>open_ed-424</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG =</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>hi_IN</td>
<td>pciscii806</td>
<td>latin1</td>
<td>open_ed-113 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>hr_BA</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Croatian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>hr_HR</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Croatian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>hu_HU</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Hungarian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>id_ID</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>is_IS</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 8</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_IT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 8</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ja_JP</td>
<td>shift-jis</td>
<td>euc-jp, shift-jis¹</td>
<td>ibm-939</td>
<td>Japanese</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kk_KZ</td>
<td>kz1048</td>
<td>Cyrillic</td>
<td>open_ed-116 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>kl_GL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ko_KR</td>
<td>euc-kr</td>
<td>euc-kr</td>
<td>ibm-933</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kw_GB</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-114 8</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>lt_LT</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-111 2</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>lv_LV</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-111 2</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>mk_MK</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-115 4</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>mr_IN</td>
<td>pciscii806</td>
<td>latin1</td>
<td>open_ed-113 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ms_MY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>mt_MT</td>
<td>wlatin1</td>
<td>latin3</td>
<td>open_ed-905</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG =</td>
<td>DATESTYLE =</td>
<td>PAPERSIZE =</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>nb_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 2</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_BE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-114 8</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_NL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-114 0</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nn_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 2</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>no_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 2</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>pl_PL</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Polish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>pt_BR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-275</td>
<td>Portugese</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>pt_PT</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-114 0</td>
<td>Portugese</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ro_RO</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ru_RU</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-102 5</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ru_UA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-115 4</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_BA</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_ME</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_RS</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sk_SK</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sl_SI</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Slovenian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sq_AL</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-115 3</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sr_BA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-102 5</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sr_ME</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-102 5</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sr_RS</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-102 5</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG =</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>sv_SE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-114 3</td>
<td>Swedish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ta_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>te_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-104 7</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>th_TH</td>
<td>pcoem87 4</td>
<td>thai</td>
<td>open_ed-116 0</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>tr_TR</td>
<td>wturkish</td>
<td>latin5</td>
<td>open_ed-102 6</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>uk_UA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-102 5</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>vi_VN</td>
<td>wvietnamese</td>
<td>latin1</td>
<td>open_ed-116 4</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_CN</td>
<td>euc-cn</td>
<td>euc-cn</td>
<td>ibm-935</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_HK</td>
<td>ms-950</td>
<td>euc-tw, ms-950¹</td>
<td>ibm-937</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_MO</td>
<td>ms-950</td>
<td>euc-tw, ms-950¹</td>
<td>ibm-937</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_SG</td>
<td>euc-cn</td>
<td>euc-cn</td>
<td>ibm-935</td>
<td>Locale</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_TW</td>
<td>ms-950</td>
<td>euc-tw, ms-950¹</td>
<td>ibm-937</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
</tbody>
</table>

¹ depends on the platform
You can use the DBCSLANG= and DBCSTYPE= system options to specify the DBCS encoding values for a SAS session. However, the ENCODING= system option is the recommended method in setting a SAS session for DBCS. The ENCODING= system option has priority. For more information, see “Setting the Encoding of a SAS Session” on page 21. Please see the “ENCODING System Option: UNIX, Windows, and z/OS” on page 694 for more information.

The following table shows the supported values for the DBCSLANG= and DBCSTYPE= system options under the z/OS, UNIX, and Windows operating environments.

**Note:** If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks.

<table>
<thead>
<tr>
<th>DBCSLANG=</th>
<th>z/OS DBCSTYPE=</th>
<th>UNIX DBCSTYPE=</th>
<th>Windows DBCSTYPE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>DBCSLANG=</td>
<td>z/OS DBCSTYPE=</td>
<td>UNIX DBCSTYPE=</td>
<td>Windows DBCSTYPE=</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>euc</td>
<td>not applicable</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Japanese</td>
<td>pcibm</td>
<td>pcibm</td>
<td>pcibm</td>
</tr>
<tr>
<td>Japanese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>not applicable</td>
<td>sjis</td>
<td>sjis</td>
</tr>
<tr>
<td>Korean</td>
<td>ibm</td>
<td>pcibm</td>
<td>pcms</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>dec</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>euc</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>pcibm</td>
<td>pcibm</td>
<td>pcibm</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
</tbody>
</table>
Encoding Values in SAS Language Elements

Overview to SAS Language Elements That Use Encoding Values

When the encoding of the SAS session is different from the encoding of the SAS file or from the data that resides in the SAS file, transcoding must occur. Consider a SAS file that was created in the Western Latin1 encoding, then moved to an IBM mainframe that uses the German EBCDIC encoding. In order for the IBM mainframe to successfully access the file, the SAS data file must be transcoded from the Western Latin1 encoding to the German EBCDIC encoding. For information about transcoding concepts, including SAS language elements that contain options for transcoding, see Chapter 4, “Transcoding for NLS,” on page 31.

SBCS, DBCS, and Unicode Encoding Values for Transcoding Data

The following table presents a list of SBCS, DBCS, and Unicode encoding values for transcoding data for all operating environments. The encoding values are valid for SAS language elements that contain options for transcoding.

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks.
### Table 23.1  SBCS, DBCS, and Unicode Encoding Values Used to Transcode Data

<table>
<thead>
<tr>
<th>Encoding Name</th>
<th>Short Name</th>
<th>Description</th>
<th>Maximum bytes per Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>aarabic</td>
<td>aara</td>
<td>Arabic Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>acroatian</td>
<td>acro</td>
<td>MacOS 36 Croatian</td>
<td>1</td>
</tr>
<tr>
<td>acyrillic</td>
<td>acyr</td>
<td>MacOS 7 Cyrillic</td>
<td>1</td>
</tr>
<tr>
<td>agreek</td>
<td>agrk</td>
<td>Greek Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>ahebrew</td>
<td>aheb</td>
<td>Hebrew Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aiceland</td>
<td>aice</td>
<td>Icelandic Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>any</td>
<td>anye</td>
<td>no transcoding is specified</td>
<td>1</td>
</tr>
<tr>
<td>arabic</td>
<td>arab</td>
<td>Arabic ISO</td>
<td>1</td>
</tr>
<tr>
<td>aroman</td>
<td>arom</td>
<td>Roman Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aromania</td>
<td>arma</td>
<td>MacOS 38 Romania</td>
<td>1</td>
</tr>
<tr>
<td>athai</td>
<td>atha</td>
<td>MacOS 21-Thai</td>
<td>1</td>
</tr>
<tr>
<td>aturkish</td>
<td>atur</td>
<td>Turkish Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aukrainian</td>
<td>aukr</td>
<td>Ukrainian Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>big5</td>
<td>big5</td>
<td>Traditional Chinese Big5</td>
<td>2</td>
</tr>
<tr>
<td>cyrillic</td>
<td>cyrl</td>
<td>Cyrillic ISO</td>
<td>1</td>
</tr>
<tr>
<td>dec-cn</td>
<td>zvms</td>
<td>Simplified Chinese DEC</td>
<td>4</td>
</tr>
<tr>
<td>dec-jp</td>
<td>jvms</td>
<td>Japanese DEC</td>
<td>2</td>
</tr>
<tr>
<td>dec-tw</td>
<td>yvms</td>
<td>Traditional Chinese DEC</td>
<td>4</td>
</tr>
<tr>
<td>ebcdic037</td>
<td>e037</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>---------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>ebcdic275</td>
<td>e275</td>
<td>Brazil EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic424</td>
<td>e424</td>
<td>Hebrew EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic425</td>
<td>e425</td>
<td>Arabic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic500</td>
<td>e500</td>
<td>International EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic838</td>
<td>e838</td>
<td>Thai EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic870</td>
<td>e870</td>
<td>Central European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic875</td>
<td>e875</td>
<td>Greek EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic905</td>
<td>e905</td>
<td>Latin 3 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic924</td>
<td>e924</td>
<td>European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1025</td>
<td>ecyr</td>
<td>Cyrillic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1026</td>
<td>etur</td>
<td>Turkish EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1047</td>
<td>elat</td>
<td>Western EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1097</td>
<td>e097</td>
<td>Farsi Bilingual EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1112</td>
<td>ebal</td>
<td>Baltic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1122</td>
<td>eest</td>
<td>Estonian EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1130</td>
<td>evie</td>
<td>Vietnamese EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1137</td>
<td>e137</td>
<td>Devanagari EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1140</td>
<td>e140</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1141</td>
<td>e141</td>
<td>Austria/Germany EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1142</td>
<td>e142</td>
<td>Denmark/Norway EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>ebcdic1143</td>
<td>e143</td>
<td>Finland/Sweden EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1144</td>
<td>e144</td>
<td>Italy EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1145</td>
<td>e145</td>
<td>Spain EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1146</td>
<td>e146</td>
<td>United Kingdom EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1147</td>
<td>e147</td>
<td>France EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1148</td>
<td>e148</td>
<td>International EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1149</td>
<td>e149</td>
<td>Iceland EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1153</td>
<td>e153</td>
<td>Latin 2 Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1154</td>
<td>e154</td>
<td>Cyrillic Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1155</td>
<td>e155</td>
<td>Turkey Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1156</td>
<td>e156</td>
<td>Baltic Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1157</td>
<td>e157</td>
<td>Estonia Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1158</td>
<td>e158</td>
<td>Cyrillic Ukraine Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1160</td>
<td>e160</td>
<td>cp1160 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1164</td>
<td>e164</td>
<td>cp1164 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdicany</td>
<td>eany</td>
<td>enables you to create a data set that is compatible with all EBCDIC encodings</td>
<td>1</td>
</tr>
<tr>
<td>euc-cn</td>
<td>zeuc</td>
<td>Simplified Chinese EUC</td>
<td>2</td>
</tr>
<tr>
<td>euc-jp</td>
<td>jeuc</td>
<td>Japanese EUC</td>
<td>4</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>euc-kr</td>
<td>keuc</td>
<td>Korean EUC</td>
<td>4</td>
</tr>
<tr>
<td>euc-tw</td>
<td>yeuc</td>
<td>Traditional Chinese EUC</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-cn</td>
<td>zfuj</td>
<td>Simplified Chinese FACOM</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-jp</td>
<td>jfuj</td>
<td>Japanese FACOM</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-ko</td>
<td>kfuj</td>
<td>Korean FACOM</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-tw</td>
<td>yfuj</td>
<td>Traditional Chinese FACOM</td>
<td>4</td>
</tr>
<tr>
<td>gb18030</td>
<td>gbke</td>
<td>Simplified Chinese GB18030</td>
<td>4</td>
</tr>
<tr>
<td>greek</td>
<td>grek</td>
<td>Greek ISO</td>
<td>1</td>
</tr>
<tr>
<td>hebrew</td>
<td>hebr</td>
<td>Hebrew ISO</td>
<td>1</td>
</tr>
<tr>
<td>hitachi-cn</td>
<td>zhit</td>
<td>Simplified Chinese HITAC</td>
<td>6</td>
</tr>
<tr>
<td>hitachi-jp</td>
<td>jhit</td>
<td>Japanese HITAC</td>
<td>6</td>
</tr>
<tr>
<td>hitachi-ko</td>
<td>khit</td>
<td>Korean HITAC</td>
<td>6</td>
</tr>
<tr>
<td>hitachi-tw</td>
<td>yhit</td>
<td>Traditional Chinese HITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-jp</td>
<td>jhts</td>
<td>Japanese XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-ko</td>
<td>khts</td>
<td>Korean XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-tw</td>
<td>yhts</td>
<td>Traditional Chinese XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hp15-tw</td>
<td>yhpx</td>
<td>Traditional Chinese HP15</td>
<td>2</td>
</tr>
<tr>
<td>ibm-1381</td>
<td>zpce</td>
<td>Simplified Chinese PCIBM</td>
<td>2</td>
</tr>
<tr>
<td>ibm-930</td>
<td>j930</td>
<td>Japanese katakana</td>
<td>4</td>
</tr>
<tr>
<td>ibm-933</td>
<td>kibm</td>
<td>Korean IBM</td>
<td>4</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>ibm-935</td>
<td>zibm</td>
<td>Simplified Chinese IBM</td>
<td>4</td>
</tr>
<tr>
<td>ibm-937</td>
<td>yibm</td>
<td>Traditional Chinese IBM</td>
<td>4</td>
</tr>
<tr>
<td>ibm-939</td>
<td>jibm</td>
<td>Japanese IBM</td>
<td>4</td>
</tr>
<tr>
<td>ibm-942</td>
<td>j942</td>
<td>Japanese PCIBM</td>
<td>2</td>
</tr>
<tr>
<td>ibm-949</td>
<td>kpce</td>
<td>Korean PCIBM</td>
<td>2</td>
</tr>
<tr>
<td>iso2022cncns</td>
<td>zist</td>
<td>Traditional Chinese ISO-2022</td>
<td>4</td>
</tr>
<tr>
<td>iso2022cngb</td>
<td>ziso</td>
<td>Simplified Chinese ISO-2022</td>
<td>4</td>
</tr>
<tr>
<td>iso2022jp</td>
<td>jiso</td>
<td>Japanese ISO-2022</td>
<td>8</td>
</tr>
<tr>
<td>iso2022kr</td>
<td>kiso</td>
<td>Korean ISO-2022</td>
<td>4</td>
</tr>
<tr>
<td>kz1048</td>
<td>kz48</td>
<td>Kazakh cp1048 Cyrillic</td>
<td>1</td>
</tr>
<tr>
<td>latin1</td>
<td>lat1</td>
<td>Western ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin2</td>
<td>lat2</td>
<td>Central European ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin3</td>
<td>lat3</td>
<td>Latin 3 ISO 8859/3</td>
<td>1</td>
</tr>
<tr>
<td>latin4</td>
<td>lat4</td>
<td>Latin 4 ISO 8859/4</td>
<td>1</td>
</tr>
<tr>
<td>latin5</td>
<td>lat5</td>
<td>Turkish ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin6</td>
<td>lat6</td>
<td>Baltic ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin7</td>
<td>lat7</td>
<td>Baltic Rim</td>
<td>1</td>
</tr>
<tr>
<td>latin8</td>
<td>lat8</td>
<td>Latin 8 ISO 8859/14</td>
<td>1</td>
</tr>
<tr>
<td>latin9</td>
<td>lat9</td>
<td>European ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin10</td>
<td>lat10</td>
<td>South-eastern Europe</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>macos-1</td>
<td>jmac</td>
<td>Japanese PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>macos-2</td>
<td>ymac</td>
<td>Traditional Chinese PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>macos-3</td>
<td>kmac</td>
<td>Korean PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>macos-25</td>
<td>zmac</td>
<td>Simplified Chinese PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>ms-932</td>
<td>j932</td>
<td>Japanese PCMS</td>
<td>2</td>
</tr>
<tr>
<td>ms-936</td>
<td>zwin</td>
<td>Simplified Chinese PCMS</td>
<td>2</td>
</tr>
<tr>
<td>ms-949</td>
<td>kwin</td>
<td>Korean PCMS</td>
<td>2</td>
</tr>
<tr>
<td>ms-950</td>
<td>ywin</td>
<td>Traditional Chinese PCMS</td>
<td>2</td>
</tr>
<tr>
<td>msdos720</td>
<td>p720</td>
<td>Arabic MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>msdos737</td>
<td>p737</td>
<td>Greek MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>msdos775</td>
<td>p775</td>
<td>Baltic MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-037</td>
<td>eous</td>
<td>USA Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-275</td>
<td>eobr</td>
<td>Brazil OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-424</td>
<td>eoiw</td>
<td>Hebrew OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-425</td>
<td>eoa2</td>
<td>Arabic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-838</td>
<td>eoth</td>
<td>Thai OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-870</td>
<td>eol2</td>
<td>Central European OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-875</td>
<td>eotel</td>
<td>Greek OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-905</td>
<td>eol3</td>
<td>Latin 3 Open Edition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-924</td>
<td>eolt</td>
<td>European OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>----------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>open_ed-930</td>
<td>oe30</td>
<td>Katakana OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-933</td>
<td>oe33</td>
<td>Korean OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-935</td>
<td>oe35</td>
<td>Simplified Chinese OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-937</td>
<td>oe37</td>
<td>Traditional Chinese OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-939</td>
<td>oe39</td>
<td>Japanese IBM</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-939E</td>
<td>oe9e</td>
<td>Japanese IBM Extended'</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-1025</td>
<td>eocy</td>
<td>Cyrillic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1026</td>
<td>eotr</td>
<td>Turkish OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1047</td>
<td>eol1</td>
<td>Western OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1097</td>
<td>eofa</td>
<td>Farsi Bilingual OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1112</td>
<td>eobl</td>
<td>Baltic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1122</td>
<td>eoet</td>
<td>Estonian OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1130</td>
<td>eovi</td>
<td>Vietnamese OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1140</td>
<td>eo40</td>
<td>North American OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1141</td>
<td>eo41</td>
<td>Austria/Germany OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1142</td>
<td>eo42</td>
<td>Denmark/Norway OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1143</td>
<td>eo43</td>
<td>Finland/Sweden OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>--------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>open_ed-1144</td>
<td>eo44</td>
<td>Italy OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1145</td>
<td>eo45</td>
<td>Spain OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1146</td>
<td>eo46</td>
<td>United Kingdom OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1147</td>
<td>eo47</td>
<td>France OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1148</td>
<td>eo48</td>
<td>International OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1149</td>
<td>eoIS</td>
<td>Iceland OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1153</td>
<td>eo53</td>
<td>Latin 2 Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1154</td>
<td>eo54</td>
<td>Cyrillic Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1155</td>
<td>eo55</td>
<td>Turkey Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1156</td>
<td>eo56</td>
<td>Baltic Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1157</td>
<td>eo57</td>
<td>Estonia Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1158</td>
<td>eo58</td>
<td>Cyrillic Ukraine Euro 1158 OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1160</td>
<td>eo60</td>
<td>1160 OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1164</td>
<td>eo64</td>
<td>1164 OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>---------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>open_ed-1166</td>
<td>eo66</td>
<td>1166 OpenEdition kz-cyr</td>
<td>1</td>
</tr>
<tr>
<td>pciscii806</td>
<td>p806</td>
<td>Indian PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem437</td>
<td>p437</td>
<td>USA IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem850</td>
<td>p850</td>
<td>Western IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem852</td>
<td>p852</td>
<td>Central European IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem857</td>
<td>p857</td>
<td>Turkish IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem858</td>
<td>p858</td>
<td>European IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem860</td>
<td>p860</td>
<td>Portuguese MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>pcoem862</td>
<td>p862</td>
<td>Hebrew IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem863</td>
<td>p863</td>
<td>French Canadian IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem864</td>
<td>p864</td>
<td>Arabic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem865</td>
<td>p865</td>
<td>Nordic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem866</td>
<td>p866</td>
<td>Cyrillic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem869</td>
<td>p869</td>
<td>Greek IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem874</td>
<td>p874</td>
<td>Thai IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem921</td>
<td>p921</td>
<td>Baltic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem922</td>
<td>p922</td>
<td>Estonia IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem1129</td>
<td>pvie</td>
<td>Vietnamese IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pc1098</td>
<td>po98</td>
<td>Farsi PC</td>
<td>1</td>
</tr>
<tr>
<td>roman8</td>
<td>rom8</td>
<td>HP Roman 8</td>
<td>1</td>
</tr>
<tr>
<td>shift-jis</td>
<td>sjis</td>
<td>Japanese SJIS</td>
<td>2</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>shift-jis04</td>
<td>sjs4</td>
<td>Shift_JIS-2004 surrogate pair support</td>
<td>2</td>
</tr>
<tr>
<td>thai</td>
<td>thai</td>
<td>Thai ISO</td>
<td>1</td>
</tr>
<tr>
<td>us-ascii</td>
<td>ansi</td>
<td>enables you to create a data set that is compatible with all ASCII encodings</td>
<td>1</td>
</tr>
<tr>
<td>utf-8</td>
<td>utf8</td>
<td>Unicode (UTF-8)</td>
<td>4</td>
</tr>
<tr>
<td>utf-16be</td>
<td>u16b</td>
<td>Unicode (UTF-16BE)</td>
<td>2 1</td>
</tr>
<tr>
<td>utf-16le</td>
<td>u16l</td>
<td>Unicode (UTF-16LE)</td>
<td>2 1</td>
</tr>
<tr>
<td>utf-32be</td>
<td>u32b</td>
<td>Unicode (UTF-32BE)</td>
<td>4 2</td>
</tr>
<tr>
<td>utf-32le</td>
<td>u32l</td>
<td>Unicode (UTF-32LE)</td>
<td>4 2</td>
</tr>
<tr>
<td>warabic</td>
<td>wara</td>
<td>Arabic Windows</td>
<td>1</td>
</tr>
<tr>
<td>wbaltic</td>
<td>wbal</td>
<td>Baltic Windows</td>
<td>1</td>
</tr>
<tr>
<td>wcyrillic</td>
<td>wcyr</td>
<td>Cyrillic Windows</td>
<td>1</td>
</tr>
<tr>
<td>wgreek</td>
<td>wgrk</td>
<td>Greek Windows</td>
<td>1</td>
</tr>
<tr>
<td>whebrew</td>
<td>wheb</td>
<td>Hebrew Windows</td>
<td>1</td>
</tr>
<tr>
<td>wlatin1</td>
<td>wlt1</td>
<td>Western Windows</td>
<td>1</td>
</tr>
<tr>
<td>wlatin2</td>
<td>wlt2</td>
<td>Central European Windows</td>
<td>1</td>
</tr>
<tr>
<td>wturkish</td>
<td>wtur</td>
<td>Turkish Windows</td>
<td>1</td>
</tr>
<tr>
<td>vvietnamese</td>
<td>vvie</td>
<td>Vietnamese Windows</td>
<td>1</td>
</tr>
</tbody>
</table>

1 UTF-16BE and UTF-16LE have a fixed length of two bytes per character.
2 UTF-32BE and UTF-32LE have a fixed length of four bytes per character.
UNIX Encoding Values

The encodings in the following tables are valid in UNIX environments.

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks.

<table>
<thead>
<tr>
<th>ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arabic</td>
<td>Arabic (ISO 8859-6)</td>
</tr>
<tr>
<td>cyrillic</td>
<td>Cyrillic (ISO 8859-5)</td>
</tr>
<tr>
<td>greek</td>
<td>Greek (ISO 8859-7)</td>
</tr>
<tr>
<td>hebrew</td>
<td>Hebrew (ISO 8859-8)</td>
</tr>
<tr>
<td>latin1</td>
<td>Western (ISO 8859-1)</td>
</tr>
<tr>
<td>latin2</td>
<td>Central Europe (ISO 8859-2)</td>
</tr>
<tr>
<td>latin5</td>
<td>Turkish (ISO 8859-9)</td>
</tr>
<tr>
<td>latin6</td>
<td>Baltic (ISO 8859-4)</td>
</tr>
<tr>
<td>Latin7</td>
<td>Baltic (ISO 8859-13)</td>
</tr>
<tr>
<td>latin8</td>
<td>Celtic (ISO 8859-14)</td>
</tr>
<tr>
<td>ENCODING= Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>latin9</td>
<td>European (ISO 8859-15)</td>
</tr>
<tr>
<td>latin10</td>
<td>South-Eastern European (ISO 8859-16)</td>
</tr>
<tr>
<td>thai</td>
<td>Thai (ISO 8859-11)</td>
</tr>
</tbody>
</table>

Table 24.2  Double-Byte Encodings for UNIX

<table>
<thead>
<tr>
<th>ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>big5</td>
<td>Traditional Chinese (Big5)</td>
</tr>
<tr>
<td>euc-cn</td>
<td>Simplified Chinese (EUC)</td>
</tr>
<tr>
<td>euc-jp</td>
<td>Japanese (EUC)</td>
</tr>
<tr>
<td>euc-kr</td>
<td>Korean (EUC)</td>
</tr>
<tr>
<td>euc-tw</td>
<td>Traditional Chinese (EUC)</td>
</tr>
<tr>
<td>shift-jis</td>
<td>Japanese (SJIS)</td>
</tr>
</tbody>
</table>

UNIX also supports the utf-8 Unicode encoding.

Windows Encoding Values

The encodings in the following tables are valid in the Windows operating environment.

Note: If an encoding-value contains a hyphen (-), enclose the encoding value in quotation marks.

Table 24.3  Single-Byte Encodings for Windows

<table>
<thead>
<tr>
<th>Description</th>
<th>Windows ENCODING= Value</th>
<th>MS-DOS ENCODING= Value</th>
<th>IBM-PC ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>warabic</td>
<td>msdos720</td>
<td>pcoem864</td>
</tr>
<tr>
<td>Baltic</td>
<td>wbaltic</td>
<td>msdos775</td>
<td>pcoem921</td>
</tr>
<tr>
<td>Description</td>
<td>Windows ENCODING= Value</td>
<td>MS-DOS ENCODING= Value</td>
<td>IBM-PC ENCODING= Value</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Central Europe</td>
<td>wlatin2</td>
<td>not applicable</td>
<td>pcoem852</td>
</tr>
<tr>
<td>Cyrillic</td>
<td>wcyrillic</td>
<td>not applicable</td>
<td>pcoem866 pcoem855</td>
</tr>
<tr>
<td>Estonia</td>
<td>wbaltic</td>
<td>not applicable</td>
<td>pcoem922</td>
</tr>
<tr>
<td>European</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem858</td>
</tr>
<tr>
<td>Farsi</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pc1098</td>
</tr>
<tr>
<td>French Canadian</td>
<td>wlatin1</td>
<td>not applicable</td>
<td>pcoem863</td>
</tr>
<tr>
<td>Greek</td>
<td>wgreek</td>
<td>msdos737</td>
<td>not applicable</td>
</tr>
<tr>
<td>Hebrew</td>
<td>whebrew</td>
<td>not applicable</td>
<td>pcoem862</td>
</tr>
<tr>
<td>Indian Script Code</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pciscii806</td>
</tr>
<tr>
<td>Nordic</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem865</td>
</tr>
<tr>
<td>Portuguese</td>
<td>wlatin1</td>
<td>pcoem860</td>
<td>not applicable</td>
</tr>
<tr>
<td>Thai</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem874</td>
</tr>
<tr>
<td>Turkish</td>
<td>wturkish</td>
<td>not applicable</td>
<td>pcoem857</td>
</tr>
<tr>
<td>USA</td>
<td>wlatin1</td>
<td>not applicable</td>
<td>pcoem437</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>wvietnamese</td>
<td>not applicable</td>
<td>not applicable</td>
</tr>
<tr>
<td>Western</td>
<td>wlatin1</td>
<td>not applicable</td>
<td>pcoem858</td>
</tr>
</tbody>
</table>

**Table 24.4  Windows Double-Byte Encodings**

<table>
<thead>
<tr>
<th>Description</th>
<th>PCMS ENCODING= Value</th>
<th>No Vendor ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional Chinese</td>
<td>ms-950</td>
<td>big5</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>ms-936</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>ms-932</td>
<td>shift-jis</td>
</tr>
</tbody>
</table>
### z/OS Encoding Values

The encodings in the following tables are valid in the z/OS operating environment.

**Note:** If an encoding-value contains a hyphen (-), enclose the encoding value in quotation marks.

**Table 24.5  Single-Byte Encodings for z/OS**

<table>
<thead>
<tr>
<th>Encoding ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC037</td>
<td>EBCDIC cp037- Old North America</td>
</tr>
<tr>
<td>EBCDIC275</td>
<td>EBCDIC cp275-Brazil</td>
</tr>
<tr>
<td>EBCDIC425</td>
<td>EBCDIC cp425-Arabic</td>
</tr>
<tr>
<td>EBCDIC838</td>
<td>EBCDIC cp838-Thai</td>
</tr>
<tr>
<td>EBCDIC870</td>
<td>EBCDIC cp870-Central Europe</td>
</tr>
<tr>
<td>EBCDIC875</td>
<td>EBCDIC cp875-Greek</td>
</tr>
<tr>
<td>EBCDIC905</td>
<td>EBCDIC cp905-Latin 3</td>
</tr>
<tr>
<td>EBCDIC924</td>
<td>EBCDIC cp924-Western Europe</td>
</tr>
<tr>
<td>EBCDIC1025</td>
<td>EBCDIC cp1025-Cyrillic</td>
</tr>
<tr>
<td>EBCDIC1026</td>
<td>EBCDIC cp1026-Turkish</td>
</tr>
<tr>
<td>EBCDIC1047</td>
<td>EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>EBCDIC1097</td>
<td>EBCDIC cp1097-Farsi Bilingual</td>
</tr>
</tbody>
</table>

**Note:** Windows also supports the utf-8 Unicode encoding.
<table>
<thead>
<tr>
<th>Encoding ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC1112</td>
<td>EBCDIC cp1112-Baltic</td>
</tr>
<tr>
<td>EBCDIC1122</td>
<td>EBCDIC cp1122-Estonian</td>
</tr>
<tr>
<td>EBCDIC1130</td>
<td>EBCDIC cp1130-Vietnamese</td>
</tr>
<tr>
<td>EBCDIC1137</td>
<td>EBCDIC cp1137-Devanagari</td>
</tr>
<tr>
<td>EBCDIC1140</td>
<td>EBCDIC cp1140-North America</td>
</tr>
<tr>
<td>EBCDIC1141</td>
<td>EBCDIC cp1141-German/Austrian</td>
</tr>
<tr>
<td>EBCDIC1142</td>
<td>EBCDIC cp1142-Danish/Norwegian</td>
</tr>
<tr>
<td>EBCDIC1143</td>
<td>EBCDIC cp1143-Finnish/Swedish</td>
</tr>
<tr>
<td>EBCDIC1144</td>
<td>EBCDIC cp1144-Italian</td>
</tr>
<tr>
<td>EBCDIC1145</td>
<td>EBCDIC cp1145-Spanish</td>
</tr>
<tr>
<td>EBCDIC1146</td>
<td>EBCDIC cp1146-English (UK)</td>
</tr>
<tr>
<td>EBCDIC1147</td>
<td>EBCDIC cp1147-French</td>
</tr>
<tr>
<td>EBCDIC1148</td>
<td>EBCDIC cp1148-International</td>
</tr>
<tr>
<td>EBCDIC1149</td>
<td>EBCDIC cp1149-Iceland</td>
</tr>
<tr>
<td>EBCDIC1153</td>
<td>EBCDIC cp1153-Latin 2 Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1154</td>
<td>EBCDIC cp1154-Cyrillic Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1155</td>
<td>EBCDIC cp1155-Turkey with euro</td>
</tr>
<tr>
<td>EBCDIC1156</td>
<td>EBCDIC cp1156-Baltic Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1157</td>
<td>EBCDIC cp1157-Estonia with euro</td>
</tr>
<tr>
<td>EBCDIC1158</td>
<td>EBCDIC cp1158-Cyrillic Ukraine with euro</td>
</tr>
<tr>
<td>OPEN_ED-037</td>
<td>OpenEdition EBCDIC cp037-Old North America</td>
</tr>
<tr>
<td>OPEN_ED-275</td>
<td>OpenEdition EBCDIC cp275-Brazil</td>
</tr>
<tr>
<td>OPEN_ED-425</td>
<td>OpenEdition EBCDIC cp425-Arabic</td>
</tr>
<tr>
<td>OPEN_ED-838</td>
<td>OpenEdition EBCDIC cp838-Thai</td>
</tr>
<tr>
<td>Encoding ENCODING= Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>OPEN_ED-870</td>
<td>OpenEdition EBCDIC cp870-Central Europe</td>
</tr>
<tr>
<td>OPEN_ED-875</td>
<td>OpenEdition EBCDIC cp875-Greek</td>
</tr>
<tr>
<td>OPEN_ED-905</td>
<td>OpenEdition EBCDIC cp905-Latin 3</td>
</tr>
<tr>
<td>OPEN_ED-924</td>
<td>OpenEdition EBCDIC cp924-Western Europe</td>
</tr>
<tr>
<td>OPEN_ED-1025</td>
<td>OpenEdition EBCDIC cp1025-Cyrillic</td>
</tr>
<tr>
<td>OPEN_ED-1026</td>
<td>OpenEdition EBCDIC cp1026-Turkish</td>
</tr>
<tr>
<td>OPEN_ED-1047</td>
<td>OpenEdition EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>OPEN_ED_1097</td>
<td>OpenEdition EBCDIC cp1097-Farsi Bilingual</td>
</tr>
<tr>
<td>OPEN_ED-1112</td>
<td>OpenEdition EBCDIC cp1112-Baltic</td>
</tr>
<tr>
<td>OPEN_ED-1122</td>
<td>OpenEdition EBCDIC cp1122-Estonish</td>
</tr>
<tr>
<td>OPEN_ED-1130</td>
<td>OpenEdition EBCDIC cp1130-Vietnamese</td>
</tr>
<tr>
<td>OPEN_ED-1137</td>
<td>OpenEdition EBCDIC cp1137-Devanagari</td>
</tr>
<tr>
<td>OPEN_ED-1140</td>
<td>OpenEdition EBCDIC cp1140-North America</td>
</tr>
<tr>
<td>OPEN_ED-1141</td>
<td>OpenEdition EBCDIC cp1141-German/Austrian</td>
</tr>
<tr>
<td>OPEN_ED-1142</td>
<td>OpenEdition EBCDIC cp1142-Danish/ Norwegian</td>
</tr>
<tr>
<td>OPEN_ED-1143</td>
<td>OpenEdition EBCDIC cp1143-Finnish/Swedish</td>
</tr>
<tr>
<td>OPEN_ED-1144</td>
<td>OpenEdition EBCDIC cp1144-Italian</td>
</tr>
<tr>
<td>OPEN_ED-1145</td>
<td>OpenEdition EBCDIC cp1145-Spanish</td>
</tr>
<tr>
<td>OPEN_ED-1146</td>
<td>OpenEdition EBCDIC cp1146-English (UK)</td>
</tr>
<tr>
<td>OPEN_ED-1147</td>
<td>OpenEdition EBCDIC cp1147-French</td>
</tr>
<tr>
<td>OPEN_ED-1148</td>
<td>OpenEdition EBCDIC cp1148-International</td>
</tr>
<tr>
<td>OPEN_ED-1149</td>
<td>OpenEdition EBCDIC cp1149-Iceland</td>
</tr>
<tr>
<td>OPEN_ED-1153</td>
<td>OpenEdition EBCDIC cp1153-Latin 2 Multilingual with euro</td>
</tr>
</tbody>
</table>
### Double-Byte Encodings for z/OS

<table>
<thead>
<tr>
<th>Description</th>
<th>ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>OPEN_ED-939</td>
</tr>
<tr>
<td>Korean</td>
<td>OPEN_ED-933</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>OPEN_ED-935</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>OPEN_ED-937</td>
</tr>
</tbody>
</table>
Appendixes

Appendix 1

Additional NLS Language Elements ............................................ 851

Appendix 2

Encoding and Alias Names ....................................................... 943

Appendix 3

Time Zone IDs and Time Zone Names ......................................... 953

Appendix 4

%COPY_TO_NEW_ENCODING Macro ............................................. 991
850
Additional NLS Language Elements

Dictionary

EURDFDDw. Format ........................................ 852
EURDFDEw. Format ........................................ 852
EURDFDNw. Format ......................................... 854
EURDFDTw.d Format ........................................ 855
EURDFDWNw. Format ........................................ 857
EURDFMNw. Format ......................................... 859
EURDFMYw. Format ......................................... 861
EURDFWDXw. Format ....................................... 863
EURDFWKXw. Format ....................................... 865
EURFRATS.w.d Format ...................................... 867
EURFRBEFw.d Format ....................................... 870
EURFRCHFw.d Format ....................................... 871
EURFRDEMw.d Format ....................................... 873
EURFRDKKw.d Format ...................................... 874
EURFRESpw.d Format ....................................... 875
EURFRFMw.d Format ........................................ 877
EURFRFRFw.d Format ....................................... 878
EURFRGBPw.d Format ...................................... 880
EURFRGRDw.d Format ...................................... 881
EURFRHUFw.d Format ...................................... 882
EURFRIEPw.d Format ....................................... 884
EURFRITLw.d Format ....................................... 885
EURFRLUFW.d Format ...................................... 886
EURFRNGLw.d Format ....................................... 888
EURFRNOKw.d Format ...................................... 889
EURFRPLZw.d Format ...................................... 891
EURFRPTEw.d Format ....................................... 892
EURFRROLw.d Format ...................................... 893
EURFRRUNW.d Format ..................................... 895
EURFRSEKw.d Format ...................................... 896
EURFRSITw.d Format ....................................... 897
EURFRTRLw.d Format ...................................... 899
EURTOATS.w.d Format ..................................... 900
EURTOBEFw.d Format ..................................... 901
EURTOCHFw.d Format ..................................... 903
EURTOCZKW.d Format .................................... 904
EURTODEMw.d Format ..................................... 905
EURTODKKw.d Format ..................................... 907
EURTOESPw.d Format ..................................... 908
EURTOFIMw.d Format ..................................... 910
EURTOFIMw.d Format ..................................... 911
Additional NLS Language Elements

The following EUR language elements have been replaced with NL language elements. The EUR elements are supported in SAS 9.3, but SAS recommends that you use the NL elements.

Dictionary

EURDFDDw. Format

Writes international date values in the form *dd.mm.yy* or *dd.mm.yyyy*.

**Category:** Date and Time

**Alignment:** Right

**Syntax**

`EURDFDDw.`

**Syntax Description**

*w* specifies the width of the output field.
<table>
<thead>
<tr>
<th>Default</th>
<th>8 (except Finnish, which is 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>2–10</td>
</tr>
<tr>
<td>Tip</td>
<td>When w is from 2 to 5, SAS prints as much of the month and day as possible. When w is 7, the date appears as a two-digit year without slashes, and the value is right-aligned in the output field.</td>
</tr>
</tbody>
</table>

**Details**

The EURDFDDw. format writes SAS date values in the form *dd.mm.yy* or *dd.mm.yyyy*, where

- **dd**
  
is the two-digit integer that represents the day of the month.

- **mm**
  
is the two-digit integer that represents the month.

- **yy or yyyy**
  
is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the `DFLANG=` system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: UNIX, Windows, and z/OS" on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the `DFLANG=` system option.

**Example**

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the `DFLANG=` system option is set to Spanish.

```
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the international date value. The third PUT statement uses the French language prefix in the format to write the international date value. Therefore, the value of the `DFLANG=` option is ignored.

```
options dflang=spanish;
data _null_;  
  input day;  
  put day eurdfdd8.;  
  datalines;  
    15342  
  ;
```

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfdd8.;</td>
<td>02.01.02</td>
</tr>
</tbody>
</table>
EURDFDEw. Format

Writes international date values in the form \textit{ddmmmyy} or \textit{ddmmmyyyy}.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Right</td>
</tr>
</tbody>
</table>

Syntax

\texttt{EURDFDEw.}

Syntax Description

\texttt{w}

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>7 (except Finnish)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>5–9 (except Finnish)</td>
</tr>
<tr>
<td>Note</td>
<td>If you use the Finnish (FIN) language prefix, the \texttt{w} range is 9–10 and the default is 9.</td>
</tr>
</tbody>
</table>

Details

The EURDFDEw. format writes SAS date values in the form \textit{ddmmmyy} or \textit{ddmmmyyyy}:

\begin{itemize}
  \item \texttt{dd} is an integer that represents the day of the month.
  \item \texttt{mmm} is the first three letters of the month name.
  \item \texttt{yy} or \texttt{yyyy} is a two-digit or four-digit integer that represents the year.
\end{itemize}

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: UNIX, Windows, and z/OS" on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.
Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set, such as UTF-8.

Example

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Spanish.

```sas
options dflang=spanish;

put day eurdfde9.;
put day espdfde9.;
put day fradfde9.;
datalines;
15342;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfde9.;</td>
<td>02ene2002</td>
</tr>
<tr>
<td>put date espdfde9.;</td>
<td>02ene2002</td>
</tr>
<tr>
<td>put date fradfde9.;</td>
<td>02jan2002</td>
</tr>
</tbody>
</table>

EURDFDNw. Format

Writes international date values as the day of the week.

Category: Date and Time
Alignment: Right
Syntax

EURDFDNw.

Syntax Description

w
specifies the width of the output field.

Default 1
Range 1–32

Details

The EURDFDNw. format writes SAS date values in the form **day-of-the-week**:

- **day-of-the-week** is represented as 1=Monday, 2=Tuesday, and so on.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: UNIX, Windows, and z/OS" on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

**Note:** The EUR-date formats require European character sets and encodings. Some formats work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Example

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Spanish.

```
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the day of the week in Spanish. The third PUT statement uses the Italian language prefix in the format to write the day of the week in Italian. Therefore, the value of the DFLANG= option is ignored.

```
options dflang=spanish;
data _null_;  
input day;  
put day eurdfdn.;  
put day espdfdn.;  
put day itadfdn.;  
datalines;  
15342  
;```
**EURDFDTw.d Format**

**Syntax**

**EURDFDTw.d**

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - **Default**: 16
  - **Range**: 7–40
  - **Tip**: If you want to write a SAS datetime value with the date, hour, and seconds, the width (w) must be at least 16. Add an additional two places to the width if you want to return values with optional decimal fractions of seconds.

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - **Range**: 1–39
  - **Restrictions**: must be less than w
    - If \( w - d < 17 \), SAS truncates the decimal values.

**Details**

The EURDFDTw.d format writes SAS datetime values in the form \( ddmmyy:hh:mm:ss.ss \).
is an integer that represents the day of the month.

**mmm**
is the first three letters of the month name.

**yy** or **yyyy**
is a two-digit or four-digit integer that represents the year.

**hh**
is the number of hours that range from 00 through 23.

**mm**
is the number of minutes that range from 00 through 59.

**ss.ss**
is the number of seconds that range from 00 through 59 with the fraction of a second following the decimal point.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

**Note:** The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

**Example**
The example table uses the input value of 1347453583, which is the SAS datetime value that corresponds to September 12, 2002, at 12:39:43 p.m. The first PUT statement assumes that the DFLANG= system option is set to German.

```sas
options dflang=german;

data _null_;
  input date;
  put date= ;
  put date eurdfdt20.;
  put date deudfdt20.;
  put date itadfdt20.;
  datalines;
  1347453583;

run;
```

The second PUT statement uses the German language prefix in the format to write the international datetime value in German. The third PUT statement uses the Italian language prefix in the format to write the international datetime value in Italian. The value of the DFLANG= option, therefore, is ignored.

```sas
options dflang=german;
data _null_;
  input date;
  put date eurdfdt20.;
  put date deudfdt20.;
  put date itadfdt20.;
  datalines;
  1347453583;

run;
```
EURDFDWNw. Format

Writes international date values as the name of the day.

**Category:** Date and Time

**Alignment:** Right

**Syntax**

`EURDFDWNw.`

**Syntax Description**

*w* specifies the width of the output field.

The default depends on the language prefix that you use. The following table shows the default value for each language:

<table>
<thead>
<tr>
<th>Language</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>9</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>9</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>10</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>7</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>7</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>9</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>11</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>8</td>
</tr>
<tr>
<td>Language</td>
<td>Default</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>10</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>9</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>9</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>10</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>7</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>12</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>13</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>11</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>10</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>9</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>7</td>
</tr>
<tr>
<td>Swiss-French (FRS)</td>
<td>8</td>
</tr>
<tr>
<td>Swiss-German (DES)</td>
<td>10</td>
</tr>
</tbody>
</table>

Default: depends on the language prefix you use.

Range: 1–32

Tip: If you omit w, SAS prints the entire name of the day.

Details

If necessary, SAS truncates the name of the day to fit the format width. The EURDFDWNw. format writes SAS date values in the form `day-name`:

`day-name`

is the name of the day.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.
Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Example

The following example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes that the DFLANG= system option is set to French.

```
options dflang=french;
put day eurdfdwn8.;
```

The second PUT statement uses the French language prefix in the format to write the day of the week in French. The third PUT statement uses the Spanish language prefix in the format to write the day of the week in Spanish. Therefore, the value of the DFLANG= option is ignored.

```
options dflang=french;
data _null_;  
input day;  
put day eurdfdwn8.;  
put day fradfdwn8.;  
put day espdfdwn8.;  
datalines;  
15344  
;  
run;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>put day eurdfdwn8.;</td>
<td>Vendredi</td>
</tr>
<tr>
<td>put day fradfdwn8.;</td>
<td>Vendredi</td>
</tr>
<tr>
<td>put day espdfdwn8.;</td>
<td>viernes</td>
</tr>
</tbody>
</table>

**EURDFMNw. Format**

Writes international date values as the name of the month.

Category: Date and Time

Alignment: Right
Syntax

\texttt{EURDFMNw.}

Syntax Description

\textit{w}

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>9 (except for Finnish and Spanish)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>1–32</td>
</tr>
</tbody>
</table>

\textbf{Note}  
If you use the Finnish (FIN) language prefix, the default value for \textit{w} is 11. If you use the Spanish (ESP) language prefix, the default value for \textit{w} is 10.

Details

If necessary, SAS truncates the name of the month to fit the format width. The \texttt{EURDFMNw.} format writes SAS date values in the form \textit{month-name}:

\textit{month-name}

is the name of the month.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

\textbf{Note}: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Example

The example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes that the DFLANG= system option is set to Italian.

\begin{verbatim}
options dflang=ita;

data _null_; input day; put day eurdfmn10.;
\end{verbatim}

The second PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. The third PUT statement uses German language prefix in the format to write the name of the month in German. Therefore, the value of the DFLANG= option is ignored.

\begin{verbatim}
options dflang=ita;
data _null_; input day; put day eurdfmn10.;
\end{verbatim}
put day itadfmn10.;
put day deudfmn10.;
datalines;
  15344
;
run;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfmn10.;</td>
<td>janvier</td>
</tr>
<tr>
<td>put date itadfmn10.;</td>
<td>Gennaio</td>
</tr>
<tr>
<td>put date deudfmn10.;</td>
<td>Januar</td>
</tr>
</tbody>
</table>

**EURDFMYw. Format**

Writes international date values in the form *mmyy* or *mmmyyy*.

**Category:** Date and Time

**Alignment:** Right

**Syntax**

**EURDFMYw.**

**Syntax Description**

*w*

specifies the width of the output field.

**Default**

5 (except for Finnish)

**Range**

5–7

**Note**

If you use the Finnish (FIN) language prefix, the value for *w* must be 8, which is the default value.

**Details**

The EURDFMYw. format writes SAS date values in the form *mmyy*, where

*mmmm*

is the first three letters of the month name.

*yy or yyyy*

is a two-digit or four-digit integer that represents the year.
You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Example

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Spanish.

```sas
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the name of the month in Spanish. The third PUT statement uses the French language prefix in the format to write the name of the month in French. Therefore, the value of the DFLANG= option is ignored.

```sas
options dflang=spanish;
data _null_;  
input date; 
  put date eurdfmy7.;  
  put date espdfmy7.;  
  put date fradfmy7.;
datalines;  
15342;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfmy7.;</td>
<td>ene2002</td>
</tr>
<tr>
<td>put date espdfmy7.;</td>
<td>ene2002</td>
</tr>
<tr>
<td>put date fradfmy7.;</td>
<td>jan2002</td>
</tr>
</tbody>
</table>
EURDFWDXw. Format

Writes international date values as the name of the month, the day, and the year in the form \textit{dd month-name yy} (or \textit{yyyy}).

<table>
<thead>
<tr>
<th>Language</th>
<th>Maximum</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>40</td>
<td>18</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Language</td>
<td>Maximum</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>37</td>
<td>23</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Swiss-French (FRS)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Swiss-German (DES)</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

Default: depends on the language prefix you use.

Range: 3–(maximum width)

Tip: If the value for $w$ is too small to include the complete day of the week and the month, SAS abbreviates as necessary.

Details

The EURDFWDXw. format writes SAS date values in the form `dd month-name yy` or `dd month-name yyyy`:

- `$dd$` is an integer that represents the day of the month.
- `month-name` is the name of the month.
- `yy` or `yyyy` is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: UNIX, Windows, and z/OS" on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of
certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Comparisons
The EURDFWKXw. format is the same as the EURDFWDXw. format except that EURDFWKX w. format adds the day-of-week in front of dd.

Example
The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Dutch.

```sas
options dflang=dutch;

data _null_;  
input date;  
put date eurdfwdx29.;  
put date nlddfwdx29.  
put date itadfwdx17.;  
datalines;  
15342  
;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day eurdfwdx29.;</td>
<td>2 januari 2002</td>
</tr>
<tr>
<td>put day nlddfwdx29.;</td>
<td>2 januari 2002</td>
</tr>
<tr>
<td>put day itadfwdx17.;</td>
<td>02 Gennaio 1998</td>
</tr>
</tbody>
</table>

EURDFWKXw. Format

Writes international date values as the name of the day and date in the form day-of-week, dd month-name yy (or yyyy).

Category: Date and Time
Alignment: Right
Syntax

\texttt{EURDFW\textit{w}.}

Syntax Description

\textit{w}

specifies the width of the output field.

The default depends on the language prefix that you use. The following table shows the default value for each language:

<table>
<thead>
<tr>
<th>Language</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>2</td>
<td>38</td>
<td>28</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>2</td>
<td>40</td>
<td>27</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>3</td>
<td>40</td>
<td>27</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>2</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>2</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>2</td>
<td>38</td>
<td>28</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>2</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>3</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>3</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>3</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>3</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>3</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>3</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>2</td>
<td>40</td>
<td>34</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>3</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>2</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>3</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>1</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>3</td>
<td>26</td>
<td>26</td>
</tr>
</tbody>
</table>
### Details

The EURDFWKXw. format writes SAS date values in the form *day-of-week, dd month-name yy (or yyyy)*:

- **day-of-week**: is the name of day.
- **dd**: is an integer that represents the day of the month.
- **month-name**: is the name of the month.
- **yy** or **yyyy**: is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: UNIX, Windows, and z/OS" on page 691 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

---

**Note:** The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

---

### Comparisons

The EURDFWKXw. format is the same as the EURDFWDXw. format except that EURDFWKXw. format adds *day-of-week* in front of *dd*.

---

### Example

The example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes that the DFLANG= system option is set to German.
options dflang=German;

The second PUT statement uses the German language prefix in the format to write the name of the month in German. The third PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. Therefore, the value of the DFLANG= option is ignored.

options dflang=german;
data _null_;input date;put date eurdfwkx30.;put date deudfwkx30.;put date itadfwkx17.;datalines;15344;run;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfwkx30.;</td>
<td>Freitag, 4. Januar 2002</td>
</tr>
<tr>
<td>put date deudfwkx30.;</td>
<td>Freitag, 4. Januar 2002</td>
</tr>
<tr>
<td>put date itadfwkx17.;</td>
<td>Ven, 04 Gen 2002</td>
</tr>
</tbody>
</table>

**EURFRATS w.d Format**

Converts an amount from Austrian schillings to euros.

Category: Currency Conversion

Alignment: Right

**Syntax**

\(\text{EURFRATS} w.d\)

**Syntax Description**

- \(w\) specifies the width of the output field.
  - Default: 6
  
- \(d\) specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURFRATS \( w.d \) format converts an amount from Austrian schillings to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRATS\( w.d \) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Austrian schillings, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrats5.;  
  put amount eurfrats9.2;  
  datalines;  
50  
5234.56  
52345  ;  
run;  
E4  
E3,63  
E380  
E380,41  
3.804  
E3.804,06
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrats5.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E3,63</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrats5.;</td>
<td>E380</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E380,41</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrats5.;</td>
<td>3.804</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E3.804,06</td>
</tr>
</tbody>
</table>

EURFRBEFw.d Format

Converts an amount from Belgian francs to euros.

Category: Currency Conversion

Alignment: Right
Syntax

**EURFRBEF**\(w.d\)

Syntax Description

- **w** specifies the width of the output field.
  - Default: 6

- **d** specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRBEF\(w.d\) format converts an amount from Belgian francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRBEF\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in Belgian francs, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrbef5.;  
  put amount eurfrbef9.2;  
  datalines;  
50  
5234.56  
52345  
;  
run;  
8     put amount eurfrbef5.;  
9     put amount eurfrbef9.2;  
10    datalines;  
   E1  
   E1,24  
E130  
E129,76  
1.298  
E1.297,60
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrbef5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrbef9.2;</td>
<td>E1,24</td>
</tr>
<tr>
<td>5234.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52345</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1,24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E130</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E129,76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.298</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1.297,60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amounts</td>
<td>Statements</td>
<td>Results</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrbef5.;</td>
<td>€130</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrbef9.2;</td>
<td>€129.76</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrbef5.;</td>
<td>1.298</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrbef9.2;</td>
<td>€1.297,60</td>
</tr>
</tbody>
</table>

EURFRCHFw.d Format

Converts an amount from Swiss francs to euros.

Category: Currency Conversion

Alignment: Right

Syntax

EURFRCHFw.d

Syntax Description

w
  specifies the width of the output field.
  Default  6

d
  specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRCHFw.d format converts an amount from Swiss francs to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRCHFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Swiss francs, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrchf5.;  
  put amount eurfrchf9.2;  
datalines;
SAS Log:
3       put amount eurfrchf5.;
4       put amount eurfrchf9.2;
5      datalines;
   E31
   E31,17
   E770
   E769,53
   7.695
   E7.694,94

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrchf5.;</td>
<td>E31</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E31,17</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrchf5.;</td>
<td>E770</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E769,53</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrchf5.;</td>
<td>7.695</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E7.694,94</td>
</tr>
</tbody>
</table>

**EURFRDEMW.d Format**

Converts an amount from Deutsche marks to euros.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

**EURFRDEMW.d**

**Syntax Description**

- `w` specifies the width of the output field.
  
  Default 6

- `d` specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURFRDEMw.d format converts an amount from Deutsche marks to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRDEMw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Deutsche marks, SAS statements, and the conversion results in euros.

```
data _null_;  
  input amount;  
  put amount eurfrdem5.;  
  put amount eurfrdem9.2;  
  datalines;  
50  
1234.56  
12345  
;  
run;  
8   put amount eurfrdem5.;  
9   put amount eurfrdem9.2;  
10  datalines;  
      E26  
      E25,56  
      E631  
      E631,22  
      6.312  
      E6.311,90
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrdem5.;</td>
<td>E26</td>
<td></td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrdem9.2;</td>
<td>E25,56</td>
<td></td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrdem5.;</td>
<td>E631</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurfrdem9.2;</td>
<td>E631,22</td>
<td></td>
</tr>
<tr>
<td>506.312</td>
<td>put amount eurfrdem5.;</td>
<td>E6.311</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>put amount eurfrdem9.2;</td>
<td>E6.311,90</td>
<td></td>
</tr>
</tbody>
</table>

EURFRD KKw.d Format

Converts an amount from Danish kroner to euros.
Syntax

\texttt{EURFRDKKw.d}

Syntax Description

\texttt{w} specifies the width of the output field.

Default \hspace{1cm} 6

\texttt{d} specifies the number of digits to the right of the decimal point in the numeric value.

Details

The \texttt{EURFRDKKw.d} format converts an amount from Danish kroner to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the \texttt{EURFRDKKw.d} format and the \texttt{EUROCURR} function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in Danish kroner, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrdkk5.;  
  put amount eurfrdkk9.2;  
  datalines;  
  50  
  1234.56  
  12345 ;  
run;  
SAS log:  
3 \hspace{1cm} put amount eurfrdkk5.;  
4 \hspace{1cm} put amount eurfrdkk9.2;  
5 \hspace{1cm} datalines;  
\hspace{1cm} 6,68  
\hspace{1cm} E6,68  
\hspace{1cm} E165  
\hspace{1cm} E164,83  
\hspace{1cm} 1.648  
\hspace{1cm} E1.648,18
EURFRESPw.d Format

Converts an amount from Spanish peseta to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRESPw.d

Syntax Description

w
specifies the width of the output field.
Default 6

d
specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRESPw.d format converts an amount from Spanish peseta to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRESPw.d format and the EUROCURREN function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in Spanish peseta, SAS statements, and the conversion results in euros.
data _null_;  
  input amount;  
  put amount eurfresp5.;  
  put amount eurfresp9.2;  
  datalines;  
  200  
  20234.56  
  202345  
  ;  
  run;  
  8     put amount eurfresp5.;  
  9     put amount eurfresp9.2;  
  10    datalines;  
        E1  
        E1,20  
        B122  
        E121,61  
        1.216  
        E1.216,12

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>put amount eurfresp5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfresp9.2;</td>
<td>E1,20</td>
</tr>
<tr>
<td>20234.56</td>
<td>put amount eurfresp5.;</td>
<td>E122</td>
</tr>
<tr>
<td></td>
<td>put amount eurfresp9.2;</td>
<td>E121,61</td>
</tr>
<tr>
<td>202345</td>
<td>put amount eurfresp5.;</td>
<td>1.216</td>
</tr>
<tr>
<td></td>
<td>put amount eurfresp9.2;</td>
<td>E1.216,12</td>
</tr>
</tbody>
</table>

**EURFRFIMw.d Format**

Converts an amount from Finnish markkas to euros.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

**EURFRFIMw.d**

**Syntax Description**

\( w \)  
specifies the width of the output field.
**Default**  6

**d**  
specifies the number of digits to the right of the decimal point in the numeric value.

**Details**  
The EURFRFIMw.d format converts an amount from Finnish markkas to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRFIMw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

**Example**  
The following table shows input values in Finnish markkas, SAS statements, and the conversion results in euros.

```
data _null_;  
  input amount;  
  put amount eurfrfim5.;  
  put amount eurfrfim9.2;  
  datalines;  
50  
1234.56  
12345  
;  
run;  
8    put amount eurfrfim5.;  
9    put amount eurfrfim9.2;  
10   datalines;  
   E8  
   E8,41  
   E208  
   E207,64  
2.076  
E2.076,28
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrfim5.; E8</td>
<td>E8</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfim9.2;   E8,41</td>
<td></td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrfim5.; E208</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfim9.2; E207,64</td>
<td></td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrfim5.; 2.076</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfim9.2; E2.076,28</td>
<td></td>
</tr>
</tbody>
</table>
EURFRFRFw.d Format

Converts an amount from French francs to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRFRF<sub>w.d</sub>

Syntax Description

<sub>w</sub>

specifies the width of the output field.

Default 6

<sub>d</sub>

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRFRF<sub>w.d</sub> format converts an amount from French francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRFRF<sub>w.d</sub> format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in French francs, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrfrf5.;  
  put amount eurfrfrf9.2;  
  datalines;  
  50  
  1234.56  
  12345  
;  
run;  
SAS log:  
E8  
E7,62  
E188  
E188,21  
1.882
```
EURFRGBPw.d Format

Converts an amount from British pounds to euros.

Syntax

EURFRGBPw.d

Syntax Description

w
  specifies the width of the output field.
  Default  6

d
  specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRGBPw.d format converts an amount from British pounds to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRGBPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.
Example

The following table shows input values in British pounds, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrGBP5.;  
  put amount eurfrGBP9.2;  
  datalines;  
50  
1234.56  
12345  
;  
run;  
SAS log:  
3     put amount eurfrGBP5.;  
4     put amount eurfrGBP9.2;  
5     datalines;  
    E71  
    E71.42  
1,763  
E1,763.32  
17632  
17,632.39

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrGBP5.;</td>
<td>E71</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrGBP9.2;</td>
<td>E71.42</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrGBP5.;</td>
<td>1,763</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrGBP9.2;</td>
<td>E1,763.32</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrGBP5.;</td>
<td>17632</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrGBP9.2;</td>
<td>17,632.39</td>
</tr>
</tbody>
</table>

EURFRGRDw.d Format

Converts an amount from Greek drachmas to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRGRDw.d
Syntax Description

`w`

specifies the width of the output field.

Default 6

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRGRDw.d format converts an amount from Greek drachmas to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRGRDw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in Greek drachmas, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrgrd5.;  
  put amount eurfrgrd9.2;  
  datalines;  
400  
40234.56  
402345  
;  
run;  
SAS log:  
3       put amount eurfrgrd5.;  
4       put amount eurfrgrd9.2;  
5       datalines;  
  E1  
  E1,17  
E118  
E118,03  
1.180  
E1.180,30

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>put amount eurfrgrd5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgrd9.2;</td>
<td>E1,17</td>
</tr>
<tr>
<td>40234.56</td>
<td>put amount eurfrgrd5.;</td>
<td>E118</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgrd9.2;</td>
<td>E118,03</td>
</tr>
</tbody>
</table>
EURFRHUFw.d Format

Converts an amount from Hungarian forints to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRHUFw.d

Syntax Description

w
  specifies the width of the output field.

  Default  6

d
  specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRHUFw.d format converts an amount from Hungarian forints to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRHUFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Hungarian forints, SAS statements, and the conversion results in euros.

data _null_
  input amount;
  put amount eurfrgd5.;
  put amount eurfrgd9.2;
datalines;
300
30234.56
302345
;
run;
SAS log:
3      put amount eurfrhuf5.;
4      put amount eurfrhuf9.2;
5      datalines;
   E1
   E1,15
E116
E116,14
1.161
E1.161,41

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>put amount eurfrhuf5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>E1,15</td>
</tr>
<tr>
<td>30234.56</td>
<td>put amount eurfrhuf5.;</td>
<td>E116</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>E116,14</td>
</tr>
<tr>
<td>302345</td>
<td>put amount eurfrhuf5.;</td>
<td>1.161</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>E1.161,41</td>
</tr>
</tbody>
</table>

**EURFRIEPw.d Format**

Converts an amount from Irish pounds to euros.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

**EURFRIEP**\(w,d\)

**Syntax Description**

\(w\)

specifies the width of the output field.

Default: 6

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURFRIEPw.d format converts an amount from Irish pounds to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRIEPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Irish pounds, SAS statements, and the conversion results in euros.

```sas
data _null_
  input amount;
  put amount eurfriep5.;;
  put amount eurfriep9.2;
  datalines;
1
1234.56
12345
;
run;
8       put amount eurfriep5.;;
9       put amount eurfriep9.2;
10      datalines;
   E1
   E1.27
1,568
E1,567.57
15675
15,674.92
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurfriep5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>E1.27</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfriep5.;</td>
<td>1,568</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>E1,567.57</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfriep5.;</td>
<td>15675</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>15,674.92</td>
</tr>
</tbody>
</table>

EURFRITLw.d Format

Converts an amount from Italian lire to euros.
Syntax

**EURFRITLw.d**

Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 6

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRITLw.d format converts an amount from Italian lire to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRITLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Italian lire, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfritl5.;  
  put amount eurfritl9.2;  
  datalines;  
2000
7234.56
72345
;  
run;
8       put amount eurfritl5.;  
9       put amount eurfritl9.2;  
10      datalines;
     E1
     E1,03
     E4
     E3,74
     E37
     E37,36
```
### EURFRLUFw.d Format

Converts an amount from Luxembourg francs to euros.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Currency Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Right</td>
</tr>
</tbody>
</table>

#### Syntax

**EURFRLUFw.d**

#### Syntax Description

- **w** specifies the width of the output field.
  - Default: 6
- **d** specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURFRLUFw.d format converts an amount from Luxembourg francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRLUFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

#### Example

The following table shows input values in Luxembourg francs, SAS statements, and the conversion results in euros.

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>put amount eurfritl5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfritl9.2;</td>
<td>E1,03</td>
</tr>
<tr>
<td>7234.56</td>
<td>put amount eurfritl5.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfritl9.2;</td>
<td>E3,74</td>
</tr>
<tr>
<td>72345</td>
<td>put amount eurfritl5.;</td>
<td>E37</td>
</tr>
<tr>
<td></td>
<td>put amount eurfritl9.2;</td>
<td>E37,36</td>
</tr>
</tbody>
</table>
data _null_;  
  input amount;  
  put amount eurfrluf5.;  
  put amount eurfrluf9.2;  
  datalines;  
50  
1234.56  
12345  
;  
run;  
8     put amount eurfrluf5.;  
9     put amount eurfrluf9.2;  
10    datalines;  
      E1  
      E1,24  
      E31  
      E30,60  
      E306  
      E306,02

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrluf5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E1,24</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrluf5.;</td>
<td>E31</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E30,60</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrluf5.;</td>
<td>E306</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E306,02</td>
</tr>
</tbody>
</table>

**EURFRNLGw.d Format**

Converts an amount from Dutch guilders to euros.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

`EURFRNLGw.d`

**Syntax Description**

`w`  
specifies the width of the output field.
Default  6

d
specifies the number of digits to the right of the decimal point in the numeric
value.

Details

The EURFRNLGw.d format converts an amount from Dutch guilders to an amount
in euros and produces a formatted euro value. The conversion rate is a fixed rate
that is incorporated into the EURFRNLGw.d format and the EUROCURR function.
For more information about European currency conversion and currency conversion
rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Dutch guilders, SAS statements, and the
conversion results in euros.

```
data _null_;    input amount;    put amount eurfrnlg5.;    put amount eurfrnlg9.2;   datalines; 50 1234.56 12345 ; run;  8     put amount eurfrnlg5.;  9     put amount eurfrnlg9.2; 10     datalines;     E23     E22,69    E560    E560,22    5.602    E5.601,92```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrnlg5.;</td>
<td>E23</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnlg9.2;</td>
<td>E22,69</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrnlg5.;</td>
<td>E560</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnlg9.2;</td>
<td>E560,22</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrnlg5.;</td>
<td>5.602</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnlg9.2;</td>
<td>E5.601,92</td>
</tr>
</tbody>
</table>
EURFRNOKw.d Format

Converts an amount from Norwegian krone to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRNOKw.d

Syntax Description

w
  specifies the width of the output field.
  Default 6

d
  specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRNOKw.d format converts an amount from Norwegian krone to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRNOKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Norwegian krone, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrnok5.;  
  put amount eurfrnok9.2;  
  datalines;  
  50  
  1234.56  
  12345  
;  
run;
SAS log:
  3 put amount eurfrnok5.;  
  4 put amount eurfrnok9.2;  
  5 datalines;  
  E5  
  E5,44
EURFRPLZw.d Format

Converts an amount from Polish zloty to euros.

Category: Currency Conversion

Alignment: Right

Syntax

EURFRPLZw.d

Syntax Description

w

specifies the width of the output field.

Default: 6

d

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRPLZw.d format converts an amount from Polish zloty to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRPLZw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.
Example

The following table shows input values in Polish zloty, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrplz5.;  
  put amount eurfrplz9.2;  
  datalines;  
50      
1234.56 
12345  
;  
run;  
SAS log:  
3      put amount eurfrplz5.;  
4      put amount eurfrplz9.2;  
5      datalines;  
  E12 
  E11,90 
  E294 
  E293,94 
  2.939  
  E2.939,29 

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrplz5.;</td>
<td>E12</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrplz9.2;</td>
<td>E11,90</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrplz5.;</td>
<td>E294</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrplz9.2;</td>
<td>E293,94</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrplz5.;</td>
<td>2.939</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrplz9.2;</td>
<td>E2.939,29</td>
</tr>
</tbody>
</table>

EURFRPTEw.d Format

Converts an amount from Portuguese escudos to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRPTEw.d
Syntax Description

\( w \)

specifies the width of the output field.

Default: 6

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRPTE\(w,d\) format converts an amount from Portuguese escudos to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRPTE\(w,d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Portuguese escudos, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrpte5.;  
  put amount eurfrpte9.2;  
  datalines;  
  300  
  30234.56  
  302345  
;  
run;  
8       put amount eurfrpte5.;  
9       put amount eurfrpte9.2;  
10      datalines;  
  E1  
  E1,50  
E151  
  E150,81  
  1.508  
  E1.508,09
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>put amount eurfrpte5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E1,50</td>
</tr>
<tr>
<td>30234.56</td>
<td>put amount eurfrpte5.;</td>
<td>E151</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E150,81</td>
</tr>
</tbody>
</table>
## EURFRROLw.d Format

Converts an amount from Romanian lei to euros.

**Category:** Currency Conversion  
**Alignment:** Right

### Syntax

```
EURFRROL w.d
```

### Syntax Description

- **w**  
  Specifies the width of the output field.  
  **Default:** 6

- **d**  
  Specifies the number of digits to the right of the decimal point in the numeric value.

### Details

The EURFRROLw.d format converts an amount from Romanian lei to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRROLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

### Example

The following table shows input values in Romanian lei, SAS statements, and the conversion results in euros.

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>302345</td>
<td>put amount eurfrpte5.;</td>
<td>1.508</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E1.508,09</td>
</tr>
</tbody>
</table>

```sas
data _null_;  
  input amount;  
  put amount eurfrt5.;  
  put amount eurfrt9.2;  
datalines;  
50  
5234.56  
52345  
;```
EURFRRURw.d Format

Converts an amount from Russian rubles to euros.

Category: 
Currency Conversion

Alignment: 
Right

Syntax

EURFRRURw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default \( 6 \)

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRRURw.d format converts an amount from Russian rubles to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRRURw.d format and the EUROCURR.
function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in Russian rubles, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrrur5.;  
  put amount eurfrrur9.2;  
  datalines;  
50  
5234.56  
52345  
;  
run;  
  E3  
  E2,53  
E265  
E264,80  
2.648  
E2.647,97
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrrur5.;</td>
<td>E3</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E2,53</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrrur5.;</td>
<td>E265</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E264,80</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrrur5.;</td>
<td>2.648</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E2.647,97</td>
</tr>
</tbody>
</table>

EURFRSEKw.d Format

Converts an amount from Swedish kronor to euros.

Category: Currency Conversion

Alignment: Right

Syntax

```sas
EURFRSEKW.d
```
Syntax Description

\( w \)

specifies the width of the output field.

Default 6

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRSEK\(w,d\) format converts an amount from Swedish kronor to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRSEK\(w,d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in Swedish kronor, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrsek5.;  
  put amount eurfrsek9.2;  
  datalines;  
  50  
  1234.56  
  12345  
  ;  
  run;  
  E5  
  E5,34  
  E132  
  E131,81  
  1.318  
  E1.318,08
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrsek5.;</td>
<td>E5</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E5,34</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrsek5.;</td>
<td>E132</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E131,81</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrsek5.;</td>
<td>1.318</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E1.318,08</td>
</tr>
</tbody>
</table>
EURFRSITw.d Format

Converts an amount from Slovenian tolars to euros.

Syntax

\texttt{EURFRSITw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 6

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRSIT\texttt{w.d} format converts an amount from Slovenian tolars to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRSIT\texttt{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Note: Slovenia’s currency is the Euro. The information for EURFRSIT is provided for user’s historical data.

Example

The following table shows input values in Slovenian tolars, SAS statements, and the conversion results in euros.

\begin{verbatim}
data _null_;  
  input amount;  
  put amount eurfrsit5.;  
  put amount eurfrsit9.2;  
  datalines;  
200  
20234.56  
202345  
;  
run;  
\end{verbatim}
### EURFRTRLw.d Format

Converts an amount from Turkish liras to euros.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

**EURFRTRL**<sub> w.d </sub>

#### Syntax Description

- **w**  
  Specifies the width of the output field.  
  - **Default**: 6

- **d**  
  Specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURFRTRL<sub> w.d </sub> format converts an amount from Turkish liras to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRTRL<sub> w.d </sub> format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.
Example

The following table shows input values in Turkish liras, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrtrl5.;  
  put amount eurfrtrl9.2;  
  datalines;  
  400  
  40234.56  
  402345  
  ;  
  run;  
  E1  
  E1,19  
  E119  
  E119,42  
  1.194  
  E1.194,21

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>400</td>
<td>put amount eurfrtrl5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtrl9.2;</td>
<td>E1,19</td>
</tr>
<tr>
<td>40234.56</td>
<td>put amount eurfrtrl5.;</td>
<td>E119</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtrl9.2;</td>
<td>E119,42</td>
</tr>
<tr>
<td>402345</td>
<td>put amount eurfrtrl5.;</td>
<td>1.194</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtrl9.2;</td>
<td>E1.194,21</td>
</tr>
</tbody>
</table>

EURTOATSw.d Format

Converts an amount from euros to Austrian schillings.

Category:  Currency Conversion
Alignment:  Right

Syntax

EURTOATSw.d
Syntax Description

\( w \)

specifies the width of the output field.

Default 6

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOATS\( w.d \) format converts an amount in euros to an amount in Austrian schillings. The conversion rate is a fixed rate that is incorporated into the EURTOATS\( w.d \) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Austrian schillings.

```sas
data _null_
  input amount;
  put amount eurtoats6.;
  put amount eurtoats12.2;
  datalines;
  1
  1234.56
  12345
  ;
run;
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoats6.;</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoats12.2;</td>
<td>13.76</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoats6.;</td>
<td>16988</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoats12.2;</td>
<td>16987.92</td>
</tr>
</tbody>
</table>
**Amounts | Statements | Results**
--- | --- | ---
12345 | put amount eurtoats6.; | 169871
 | put amount eurtoats12.2; | 169870.90

## EURTOBEFw.d Format

Converts an amount from euros to Belgian francs.

**Category:** Currency Conversion  
**Alignment:** Right

### Syntax

`EURTOBEFw.d`

### Syntax Description

- **w**  
  specifies the width of the output field.  
  Default: 6

- **d**  
  specifies the number of digits to the right of the decimal point in the numeric value.

### Details

The `EURTOBEFw.d` format converts an amount in euros to an amount in Belgian francs. The conversion rate is a fixed rate that is incorporated into the `EURTOBEFw.d` format and the `EUROCURR` function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

### Example

The following table shows input values in euros, SAS statements, and the conversion results in Belgian francs.

```sas
data _null_;  
  input amount;  
  put amount eurtoats6.;  
  put amount eurtoats12.2;  
  datalines;  
1  
1234.56  
12345  
;  
```

---

EURTOBEFw.d Format  
903
run;
8     put amount eurtobef6.;
9     put amount eurtobef12.2;
10    datalines;
     40
     40.34
     49802
     49802.03
     497996
     497996.07

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>put amount eurtobef6.;</td>
<td>40</td>
</tr>
<tr>
<td>40.34</td>
<td>put amount eurtobef12.2;</td>
<td>40.34</td>
</tr>
<tr>
<td>49802</td>
<td>put amount eurtobef6.;</td>
<td>49802</td>
</tr>
<tr>
<td>49802.03</td>
<td>put amount eurtobef12.2;</td>
<td>49802.03</td>
</tr>
<tr>
<td>497996</td>
<td>put amount eurtobef6.;</td>
<td>497996</td>
</tr>
<tr>
<td>497996.07</td>
<td>put amount eurtobef12.2;</td>
<td>497996.07</td>
</tr>
</tbody>
</table>

**EURTOCHFw.d Format**

Converts an amount from euros to Swiss francs.

*Category:* Currency Conversion

*Alignment:* Right

**Syntax**

```
EURTOCHFw.d
```

**Syntax Description**

- `w`
  - specifies the width of the output field.
  - Default: 6

- `d`
  - specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURTOCHFw.d format converts an amount in euros to an amount in Swiss francs. The conversion rate is a changeable rate that is incorporated into the EURTOCHFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Swiss francs.

data _null_;  
  input amount;  
  put amount eurtochf6.;  
  put amount eurtochf12.2;  
  datalines;  
  1 1234.56 12345 ;  
run;  
SAS log:  
  8       put amount eurtochf6.;  
  9       put amount eurtochf12.2;  
  10      datalines;  
          2 1.60  
          1981 1980.60  
          19805 19805.08

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtochf6.;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>put amount eurtochf12.2;</td>
<td>1.60</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtochf6.;</td>
<td>1981</td>
</tr>
<tr>
<td></td>
<td>put amount eurtochf12.2;</td>
<td>1980.60</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtochf6.;</td>
<td>19805</td>
</tr>
<tr>
<td></td>
<td>put amount eurtochf12.2;</td>
<td>19805.08</td>
</tr>
</tbody>
</table>

EURTOCZKw.d Format

Converts an amount from euros to Czech koruny.
Syntax

EURTOCZKw.d

Syntax Description

w
specifies the width of the output field.

Default 6

d
specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOCZKw.d format converts an amount in euros to an amount in Czech koruny. The conversion rate is a changeable rate that is incorporated into the EURTOCZKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Czech koruny.

data _null_;  
  input amount;  
  put amount eurtoczk6.;  
  put amount eurtoczk12.2;  
datalines;  
1  
1234.56  
12345  ;  
run;  
SAS log:  
104 put amount eurtoczk6.;  
105 put amount eurtoczk12.2;  
106 datalines;  
35  
34.86  
43032  
43032.19  
430301  
430301.02
EURTODEMw.d Format

Converts an amount from euros to Deutsche marks.

Category: Currency Conversion
Alignment: Right

Syntax

EURTODEMw.d

Syntax Description

w
   specifies the width of the output field.
   Default 6

d
   specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTODEMw.d format converts an amount in euros to an amount in Deutsche marks. The conversion rate is a fixed rate that is incorporated into the EURTODEMw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Deutsche marks.
data _null_;  
input amount;
put amount eurtodem6.;
put amount eurtodem12.2;
datalines;
1
1234.56
12345
;
run;

8       put amount eurtodem6.;
9       put amount eurtodem12.2;
10      datalines;
2
   1.96
2415
2414.59
24145
24144.72

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtodem6.;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurtodem12.2;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodem6.;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurtodem12.2;</td>
<td>1.96</td>
</tr>
</tbody>
</table>

| 1234.56 | put amount eurtodem6.;  | 2415             |
|         | put amount eurtodem12.2; | 2414.59          |

| 12345   | put amount eurtodem6.;  | 24145            |
|         | put amount eurtodem12.2; | 24144.72         |

**EURTODKKw.d Format**

Converts an amount from euros to Danish kroner.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

```
EURTODKKw.d
```

**Syntax Description**

`w`  
specifies the width of the output field.
Default  6

d  specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTODKKw.d format converts an amount in euros to an amount in Danish kroner. The conversion rate is a changeable rate that is incorporated into the EURTODKKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Danish kroner.

```
data _null_;  
  input amount;  
  put amount eurtodkk6.;  
  put amount eurtodkk12.2;  
  datalines;  
1  
1234.56  
12345  
;  
run;  
SAS log:  
62      put amount eurtodkk6.;  
63      put amount eurtodkk12.2;  
64      datalines;  
7  
9247  
9246.97  
92465  
92465.16
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtodkk6.;</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>7.49</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtodkk6.;</td>
<td>9247</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>9246.97</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtodkk6.;</td>
<td>92465</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>92465.16</td>
</tr>
</tbody>
</table>
EURTOESPw.d Format

Converts an amount from euros to Spanish peseta.

Category: Currency Conversion
Alignment: Right

Syntax

`EURTOESPw.d`

Syntax Description

`w`

specifies the width of the output field.

Default: 6

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOESPw.d format converts an amount in euros to an amount in Spanish peseta. The conversion rate is a fixed rate that is incorporated into the EURTOESPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Spanish peseta.

```
data _null_;  
  input amount;  
  put amount eurtoesp8.;  
  put amount eurtoesp12.2;  
  datalines;  
1  
1234.56  
12345  
;  
run;  
26      put amount eurtoesp8.;  
27      put amount eurtoesp12.2;  
28      datalines;  
166  
166.39  
205414
```
### EURTOFIMw.d Format

Converts an amount from euros to Finnish markkas.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

**EURTOFIMw.d**

**Syntax Description**

- **w**  
  specifies the width of the output field.  
  Default 6

- **d**  
  specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTOFIMw.d format converts an amount in euros to an amount in Finnish markkas. The conversion rate is a fixed rate that is incorporated into the EURTOFIMw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.
Example
The following table shows input values in euros, SAS statements, and the conversion results in Finnish markkas.

```
data _null_;  
  input amount;  
  put amount eurtofim6.;  
  put amount eurtofim12.2;  
  datalines;  
  1  
  1234.56  
  12345  
  ;  
  run;  
  8  put amount eurtofim6.;  
  9  put amount eurtofim12.2;  
  10  datalines;  
  ;  
  6  
  5.95  
  7340  
  7340.36  
  73400  
  73400.04
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtofim6.; put amount eurtofim12.2;</td>
<td>6 5.95</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtofim6.; put amount eurtofim12.2;</td>
<td>7340 7340.36</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtofim6.; put amount eurtofim12.2;</td>
<td>73400 73400.04</td>
</tr>
</tbody>
</table>

EURTOFRFw.d Format
Converts an amount from euros to French francs.

Category: Currency Conversion
Alignment: Right

Syntax
```
EURTOFRFw.d
```
Syntax Description

**w**
specifies the width of the output field.

Default: 6

**d**
specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOFRFw.d format converts an amount in euros to an amount in French francs. The conversion rate is a fixed rate that is incorporated into the EURTOFRFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in French francs.

data _null_;  
  input amount;  
  put amount eurtofrf6.;  
  put amount eurtofrf12.2;  
  datalines;  
1  1234.56  
2 12345  
;  
run;  
8  put amount eurtofrf6.;  
9  put amount eurtofrf12.2;  
10 datalines;  
7  
6.56  
8098  
8098.18  
80978  
80977.89

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtofrf6.;</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>6.56</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtofrf6.;</td>
<td>8098</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>8098.18</td>
</tr>
</tbody>
</table>
EURTOGBPw.d Format

Converts an amount from euros to British pounds.

Category: Currency Conversion
Alignment: Right

Syntax

EURTOGBPw.d

Syntax Description

w
  specifies the width of the output field.
  Default 6

d
  specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOGBPw.d format converts an amount in euros to an amount in British pounds. The conversion rate is a changeable rate that is incorporated into the EURTOGBPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in British pounds.

data _null_;  
  input amount;  
  put amount eurtogbp6.;  
  put amount eurtogbp12.2;  
datalines;  
1 1234.56 12345
;

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>put amount eurtogbp6.;</td>
<td>80978</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>80977.89</td>
</tr>
</tbody>
</table>
run;
SAS log:
8     put amount eurtogbp6.;
9     put amount eurtogbp12.2;
10    datalines;
1
  0.70
 864
  864.35
 8643
  8643.13

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtogbp6.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>0.70</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtogbp6.;</td>
<td>864</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>864.35</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtogbp6.;</td>
<td>8643</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>8643.13</td>
</tr>
</tbody>
</table>

**EURTOGRDw.d Format**

Converts an amount from euros to Greek drachmas.

Category: Currency Conversion

Alignment: Right

**Syntax**

`EURTOGRDw.d`

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 6

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURTOGRDw.d format converts an amount in euros to an amount in Greek drachmas. The conversion rate is a fixed rate that is incorporated into the EURTOGRDw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Greek drachmas.

data _null_;  
  input amount;  
  put amount eurtogr8.;  
  put amount eurtogr16.2;  
  datalines;  
1  
1234.56  
12345  
;  
run;  
SAS log:  
65 put amount eurtogr8.;  
66 put amount eurtogr16.2;  
67 datalines;  
341  
340.89  
420843  
420842.99  
4208225  
4208225.33

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtogr8.;</td>
<td>341</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogr16.2;</td>
<td>340.89</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtogr8.;</td>
<td>420843</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogr16.2;</td>
<td>420842.99</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtogr8.;</td>
<td>4208225</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogr16.2;</td>
<td>4208225.33</td>
</tr>
</tbody>
</table>

EURTOHUFw.d Format

Converts an amount from euros to Hungarian forints.
Syntax

**EURTOHUFw.d**

Syntax Description

\(w\)

specifies the width of the output field.

Default \(6\)

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOHUFw.d format converts an amount in euros to an amount in Hungarian forints. The conversion rate is a changeable rate that is incorporated into the EURTOHUFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Hungarian forints.

```sas
data _null_;
  input amount;
  put amount eurtohuf8.;
  put amount eurtohuf14.2;
  datalines;
1
1234.56
12345
;
run;
SAS log:
140     put amount eurtohuf8.;
141     put amount eurtohuf14.2;
142     datalines;
260     260.33
321387   321386.83
3213712  3213712.13
```
EURTOIEP\(w.d\) Format

Converts an amount from euros to Irish pounds.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

\[
\text{EURTOIEP}w.d
\]

**Syntax Description**

\(w\)

- Specifies the width of the output field.

  - Default: 6

\(d\)

- Specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOIEP\(w.d\) format converts an amount in euros to an amount in Irish pounds. The conversion rate is a fixed rate that is incorporated into the EURTOIEP\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

**Example**

The following table shows input values in euros, SAS statements, and the conversion results in Irish pounds.
data _null_;  
    input amount;  
    put amount eurtoiep6.;  
    put amount eurtoiep12.2;  
    datalines;  
 1  
1234.56  
12345  
;  
run;  
8          put amount eurtoiep6.;  
9          put amount eurtoiep12.2;  
10         datalines;  
1          0.79  
972        972.30  
9722       9722.48  

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoiep6.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>0.79</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoiep6.;</td>
<td>972</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>972.30</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoiep6.;</td>
<td>9722</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>9722.48</td>
</tr>
</tbody>
</table>

**EURTOITLw.d Format**

Converts an amount from euros to Italian lire.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

greece Юрггб

greek Syntax Description

\( w \)

specifies the width of the output field.
Default 6

d specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOITLw.d format converts an amount in euros to an amount in Italian lire. The conversion rate is a fixed rate that is incorporated into the EURTOITLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Italian lire.

data _null_;  
  input amount;  
  put amount eurtoitl8.;  
  put amount eurtoitl12.2;  
  datalines;  
  1  
  1234.56  
  12345  
  ;  
  run;  
44 put amount eurtoitl8.;  
45 put amount eurtoitl12.2;  
46 datalines;  
  1936  
  1936.27  
  2390441  
  2390441.49  
  23903253  
  23903253.15

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoitl8.;</td>
<td>1936</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoitl12.2;</td>
<td>1936.27</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoitl8.;</td>
<td>2390441</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoitl12.2;</td>
<td>2390441.49</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoitl8.;</td>
<td>23903253</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoitl12.2;</td>
<td>23903253.15</td>
</tr>
</tbody>
</table>
EURTOLUFw.d Format

Converts an amount from euros to Luxembourg francs.

Category: Currency Conversion

Syntax

**EURTOLUFw.d**

Syntax Description

w

specifies the width of the output field.

Default 6

d

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOLUFw.d format converts an amount in euros to an amount in Luxembourg francs. The conversion rate is a fixed rate that is incorporated into the EURTOLUFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Luxembourg francs.

```sas
data _null_;  
  input amount;
  put amount eurtoluf6.;
  put amount eurtoluf12.2;
  datalines;
  1
  1234.56
  12345
  ;
run;
```

```
8        put amount eurtoluf6.;
9        put amount eurtoluf12.2;
10       datalines;
       40
       40.34
     49802
```
### EURTONLGGw.d Format

Converts an amount from euros to Dutch guilders.

**Category:** Currency Conversion

**Alignment:** Right

#### Syntax

EURTONLGGw.d

#### Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 6

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTONLGGw.d format converts an amount in euros to an amount in Dutch guilders. The conversion rate is a fixed rate that is incorporated into the EURTONLGGw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.
Example

The following table shows input values in euros, SAS statements, and the conversion results in Dutch guilders.

```
data _null_;  
  input amount;  
  put amount eurtonlg6.;  
  put amount eurtonlg12.2;  
  datalines;  
1  
1234.56  
12345  
;  
run;  
8  
  put amount eurtonlg6.;  
9  
  put amount eurtonlg12.2;  
10  
  datalines;  
2  
  2.20  
  2721  
  2720.61  
27205  
  27204.80
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtonlg6.; put amount eurtonlg12.2;</td>
<td>2 2.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtonlg6.; put amount eurtonlg12.2;</td>
<td>2721 2720.61</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtonlg6.; put amount eurtonlg12.2;</td>
<td>27205 27204.80</td>
</tr>
</tbody>
</table>

EURTONOKw.d Format

Converts an amount from euros to Norwegian krone.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

```
EURTONOKw.d
```
Syntax Description

\( w \)

specifies the width of the output field.

Default 6

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTONOK\( w,d \) format converts an amount in euros to an amount in Norwegian krone. The conversion rate is a changeable rate that is incorporated into the EURTONOK\( w,d \) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Norwegian krone.

```sas
data _null_;  
    input amount; 
    put amount eurtonok6.; 
    put amount eurtonok12.2; 
    datalines; 
1  
1234.56  
12345  
; run; 
SAS log: 
158     put amount eurtonok6.;  
159     put amount eurtonok12.2;  
160     datalines;  
  9  
  9.20  
11355  
11355.11  
113546  
113545.61
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtonok6.;</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2;</td>
<td>9.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtonok6.;</td>
<td>11355</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2;</td>
<td>11355.11</td>
</tr>
</tbody>
</table>
EURTOPLZw.d Format

Converts an amount from euros to Polish zloty.

Category: Currency Conversion
Alignment: Right

Syntax

EURTOPLZw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 6

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOPLZw.d format converts an amount in euros to an amount in Polish zloty. The conversion rate is a changeable rate that is incorporated into the EURTOPLZw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Polish zloty.

data _null_
  input amount;
  put amount eurtonok6.;
  put amount eurtonok12.2;
datalines;
1 1234.56 12345
1 13546 113545.61

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>put amount eurtonok6.</td>
<td>113546</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2</td>
<td>113545.61</td>
</tr>
</tbody>
</table>
run;
SAS log:
80      put amount eurtoplz6.;
81      put amount eurtoplz12.2;
82      datalines;
   4
     4.20
5185
  5185.15
51849
51849.00

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoplz6.;</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplz12.2;</td>
<td>4.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoplz6.;</td>
<td>5185</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplz12.2;</td>
<td>5185.15</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoplz6.;</td>
<td>51849</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplz12.2;</td>
<td>51849.00</td>
</tr>
</tbody>
</table>

**EURTOPTEw.d Format**

Converts an amount from euros to Portuguese escudos.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

`EURTOPTEw.d`

**Syntax Description**

`w`  
specifies the width of the output field.  
Default: 6

`d`  
specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURTOPTEw.d format converts an amount in euros to an amount in Portuguese escudos. The conversion rate is a fixed rate that is incorporated into the EURTOPTEw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Portuguese escudos.

```sas
data _null_
  input amount;
  put amount eurtopte8.;
  put amount eurtopte12.2;
  datalines;
1
1234.56
12345
;
run;
26      put amount eurtopte8.;
27      put amount eurtopte12.2;
28      datalines;
200
200.48
247507
247507.06
2474950
2474950.29
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtopte8.;</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>put amount eurtopte12.2;</td>
<td>200.48</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtopte8.;</td>
<td>247507</td>
</tr>
<tr>
<td></td>
<td>put amount eurtopte12.2;</td>
<td>247507.06</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtopte8.;</td>
<td>2474950</td>
</tr>
<tr>
<td></td>
<td>put amount eurtopte12.2;</td>
<td>2474950.29</td>
</tr>
</tbody>
</table>

EURTOROLw.d Format

Converts an amount from euros to Romanian lei.
Syntax

**EURTOROL**\textit{w.d}

Syntax Description

\textit{w} specifies the width of the output field.

Default: 6

\textit{d} specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOROL\textit{w.d} format converts an amount in euros to an amount in Romanian lei. The conversion rate is a changeable rate that is incorporated into the EURTOROL\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Romanian lei.

```sas
data _null_;  
  input amount;  
  put amount eurtorol6.;  
  put amount eurtorol12.2;  
  datalines;  
1 1234.56 12345 ;  
run;  
SAS log:  
98 99 100  
put amount eurtorol6.; put amount eurtorol12.2; datalines;  
14 16926 16925.82 169250 169249.95
```
### EURTORURw.d Format

Converts an amount from euros to Russian rubles.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

```
EURTORURw.d
```

#### Syntax Description

- `w` specifies the width of the output field.  
  - **Default** 6

- `d` specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTORURw.d format converts an amount in euros to an amount in Russian rubles. The conversion rate is a changeable rate that is incorporated into the EURTORURw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

#### Example

The following table shows input values in euros, SAS statements, and the conversion results in Russian rubles.

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtorol6.;</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorol12.2;</td>
<td>13.71</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtorol6.;</td>
<td>16926</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorol12.2;</td>
<td>16925.82</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtorol6.;</td>
<td>169250</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorol12.2;</td>
<td>169249.95</td>
</tr>
</tbody>
</table>
data _null_;  
  input amount;  
  put amount eurtorur6.;  
  put amount eurtorur12.2;  
  datalines;  
  1  
  1234.56  
  12345  
  ;  
  run;  
SAS log:  
8       put amount eurtorur6.;  
9       put amount eurtorur12.2;  
10      datalines;  
20       19.77  
24405  
24404.78  
244036  
244035.96  

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1</td>
<td>put amount eurtorur6.;</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>19.77</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtorur6.;</td>
<td>24405</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>24404.78</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtorur6.;</td>
<td>244036</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>244035.96</td>
</tr>
</tbody>
</table>

EURTOSEKw.d Format

Converts an amount from euros to Swedish kronor.

Category: Currency Conversion
Alignment: Right

Syntax

EURTOSEKw.d

Syntax Description

w
  specifies the width of the output field.
Default 6

\(d\) specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOSEK\(w.d\) format converts an amount in euros to an amount in Swedish kronor. The conversion rate is a changeable rate that is incorporated into the EURTOSEK\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Swedish kronor.

```
data _null_;  
  input amount;  
  put amount eurtosek6.;  
  put amount eurtosek12.2;  
  datalines;  
1 1234.56  
12345  
;  
run;  
SAS log:  
86  put amount eurtosek6.;  
87  put amount eurtosek12.2;  
88  datalines;  
9  
9.37  
11563  
11562.78  
115622  
115622.16
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtosek6.; put amount eurtosek12.2;</td>
<td>9 9.37</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtosek6.; put amount eurtosek12.2;</td>
<td>11563 11562.78</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtosek6.; put amount eurtosek12.2;</td>
<td>115622 115622.16</td>
</tr>
</tbody>
</table>
EURTOSIT\(w.d\) Format

Converts an amount from euros to Slovenian tolars.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

\[ \text{EURTOSIT}w.d \]

**Syntax Description**

\(w\)

- specifies the width of the output field.  
  
  Default 6

\(d\)

- specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOSIT\(w.d\) format converts an amount in euros to an amount in Slovenian tolars. The conversion rate is a changeable rate that is incorporated into the EURTOSIT\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "Currency Representation" on page 78.

**Note:** Slovenia's currency is the Euro. The information for EURTOSIT is provided for user's historical data.

**Example**

The following table shows input values in euros, SAS statements, and the conversion results in Slovenian tolars.

```sas
data _null_;  
  input amount;  
  put amount eurtosit8.;  
  put amount eurtosit14.2;  
  datalines;  
1  
1234.56  
12345  
;  
run;  
SAS log:
```
EURTOTRLw.d Format

Converts an amount from euros to Turkish liras.

Category: Currency Conversion
Alignment: Right

Syntax

\texttt{EURTOTRLw.d}

Syntax Description

\texttt{w}
\begin{itemize}
  \item specifies the width of the output field.
\end{itemize}
\begin{itemize}
  \item Default \ 6
\end{itemize}
\texttt{d}
\begin{itemize}
  \item specifies the number of digits to the right of the decimal point in the numeric value.
\end{itemize}
Details

The EURTOTRLw.d format converts an amount in euros to an amount in Turkish liras. The conversion rate is a changeable rate that is incorporated into the EURTOTRLw.d format and the EUROCURRE function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 78.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Turkish liras.

data _null_;  
   input amount;  
   put amount eurtotrl8.;  
   put amount eurtotrl14.2;  
   datalines;  
1  1234.56  415938  415938.08  4159179  4159178.64

SAS log:
62      put amount eurtotrl8.;  
63      put amount eurtotrl14.2;  
64      datalines;
   337  336.91  415938  415938.08  4159179  4159178.64

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtotrl8.;</td>
<td>337</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>336.91</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtotrl8.;</td>
<td>415938</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>415938.08</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtotrl8.;</td>
<td>4159179</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>4159178.64</td>
</tr>
</tbody>
</table>

EURDFDEw. Informat

Reads international date values.
Syntax

**EURDFDEw.**

Required Argument

**w**

specifies the width of the input field.

Default  7 (except Finnish)

Range   7–32 (except Finnish)

Note  If you use the Finnish (FIN) language prefix, the w range is 10–32 and the default w is 10.

Details

The date values must be in the form *ddmmmyy* or *ddmmmyyyy*:

*dd*  is an integer from 01–31 that represents the day of the month.

*mmm*  is the first three letters of the month name.

*yy* or *yyyy*  is a two-digit or four-digit integer that represents the year.

You can place blanks and other special characters between day, month, and year values.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: UNIX, Windows, and z/OS" on page 691 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the DFLANG= system option.

Example

This INPUT statement uses the value of the DFLANG= system option to read the international date values in Spanish.

```sas
options dflang=spanish;
input day eurdfde10.;
```

This INPUT statement uses the Spanish language prefix in the informat to read the international date values in Spanish. The value of the DFLANG= option, therefore, is ignored.

```sas
input day espdfde10.;
```

```sas
options dflang=spanish;
data _null_;
EURDFDTw. Informat

Reads international datetime values in the form `ddmmmyy hh:mm:ss.ss` or `ddmmmyyyy hh:mm:ss.ss`.

**Category:** Date and Time

**Syntax**

`EURDFDTw.`

**Syntax Description**

`w`

specifies the width of the input field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>13–40</td>
</tr>
</tbody>
</table>

**Details**

The date values must be in the form `ddmmmyy` or `ddmmmyyyy`, followed by a blank or special character, and then the time values as `hh:mm:ss.ss`. The syntax for the date is represented as follows:

- `dd` is an integer from 01–31 that represents the day of the month.
- `mmm` is the first three letters of the month name.
- `yy` or `yyyy` is a two-digit or four-digit integer that represents the year.

The syntax for time is represented as follows:
\[ hh \]
- is the number of hours ranging from 00–23,

\[ mm \]
- is the number of minutes ranging from 00–59,

\[ ss.ss \]
- is the number of seconds ranging from 00–59 with the fraction of a second following the decimal point.

The EURDFDTw. informat requires values for both the date and the time. However, the \[ ss.ss \] portion is optional.

**Note:** SAS interprets a two-digit year as belonging to the 100-year span that is defined by the \texttt{YEARCUTOFF=} system option.

You can set the language for the SAS session with the \texttt{DFLANG=} system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “\texttt{DFLANG= System Option: UNIX, Windows, and z/OS}” on page 691 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the \texttt{DFLANG=} system option.

**Example**

This \texttt{INPUT} statement uses the value of the \texttt{DFLANG=} system option to read the international datetime values in German.

```sas
options dflang=german;
input date eurdfdt20.;
```

This \texttt{INPUT} statement uses the German language prefix to read the international datetime values in German. The value of the \texttt{DFLANG=} option, therefore, is ignored.

```sas
input date deudfdt20.;
options dflang=german;
data _null_; input date eurdfdt20.; put date; datalines;
23dez99:10:03:17.2
23dez1999:10:03:17.2
;```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>23dez99:10:03:17.2</td>
<td>1261562597.2</td>
</tr>
<tr>
<td>23dez1999:10:03:17.2</td>
<td>1261562597.2</td>
</tr>
</tbody>
</table>
EURDFMYw. Informat
Reads month and year date values in the form *mmmy* or *mmmyyyy*.

**Category:** Date and Time

**Syntax**

```
EURDFMYw.
```

**Syntax Description**

- **w** specifies the width of the input field.

  **Default** 5 (except Finnish)

  **Range** 5–32 (except Finnish)

  **Note** If you use the Finnish (FIN) language prefix, the *w* range is 7–32 and the default value for *w* is 7.

**Details**

The date values must be in the form *mmmy* or *mmmyyyy*:

- **mmm** is the first three letters of the month name.
- **yy** or **yyyy** is a two-digit or four-digit integer that represents the year.

You can place blanks and other special characters between day, month, and year values. A value that is read with EURDFMYw. results in a SAS date value that corresponds to the first day of the specified month.

**Note:** SAS interprets a two-digit year as belonging to the 100-year span that is defined by the YEARCUTOFF= system option.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 691 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the DFLANG= option.

**Example**

This INPUT statement uses the value of DFLANG= system option to read the international date values in French.
options dflang=french;
input month eurdfmy7.;

The second INPUT statement uses the French language prefix, and DFLANG is not specified.

input month fradfmy7.;
options dflang=french;
data _null_;  
input month eurdfmy7.;  
put month;
datalines;
avr1999
avr 99
;
options dflang=english;
data _null_;  
input month fradfmy7.;  
put month;
datalines;
avr1999
avr 99
;

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>--------+---------</td>
<td></td>
</tr>
<tr>
<td>avr1999</td>
<td>14335</td>
</tr>
<tr>
<td>avr 99</td>
<td>14335</td>
</tr>
</tbody>
</table>

EUROCURR Function

Converts one European currency to another.

Category: Currency Conversion

Syntax

\[
\text{EUROCURR}(\text{from-currency-amount}, \text{from-currency-code}, \text{to-currency-code})
\]

Required Arguments

- **from-currency-amount**: is a numeric value that specifies the amount to convert.
- **from-currency-code**: specifies a three-character currency code that identifies the currency that you are converting from. (See European Currency and Currency Codes on page 940.)
If `from-currency-code` has a blank value, EUROCURR converts currency values from euros to the currency of the European country that you specify.

See “Example 4: Converting Currency When One Variable Is Blank” on page 942.

**to-currency-code**

specifies a three-character currency code that identifies the currency that you are converting to. (See European Currency and Currency Codes on page 940.)

If `to-currency-code` has a blank value, EUROCURR converts values from the currency of the European country that you specify to euros.

**Details**

The following table lists European currencies and the associated currency codes. Use the currency codes to identify the type of currency that you are converting to or converting from. Several countries use the Euro as their currency instead of the currency listed in the following table. This information is provided in order to satisfy user’s historical data.

**Table A12.1 European Currency and Currency Codes**

<table>
<thead>
<tr>
<th>Currency</th>
<th>Currency code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austrian schilling</td>
<td>ATS</td>
</tr>
<tr>
<td>Belgian franc</td>
<td>BEF</td>
</tr>
<tr>
<td>British pound sterling</td>
<td>GBP</td>
</tr>
<tr>
<td>Czech koruna</td>
<td>CZK</td>
</tr>
<tr>
<td>Danish krone</td>
<td>DKK</td>
</tr>
<tr>
<td>Deutsche mark</td>
<td>DEM</td>
</tr>
<tr>
<td>Dutch guilder</td>
<td>NLG</td>
</tr>
<tr>
<td>Euro</td>
<td>EUR</td>
</tr>
<tr>
<td>Finnish markka</td>
<td>FIM</td>
</tr>
<tr>
<td>French franc</td>
<td>FRF</td>
</tr>
<tr>
<td>Greek drachma</td>
<td>GRD</td>
</tr>
<tr>
<td>Hungarian forint</td>
<td>HUF</td>
</tr>
<tr>
<td>Irish pound</td>
<td>IEP</td>
</tr>
<tr>
<td>Currency</td>
<td>Currency code</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Italian lira</td>
<td>ITL</td>
</tr>
<tr>
<td>Luxembourg franc</td>
<td>LUF</td>
</tr>
<tr>
<td>Norwegian krone</td>
<td>NOK</td>
</tr>
<tr>
<td>Polish zloty</td>
<td>PLZ</td>
</tr>
<tr>
<td>Portuguese escudo</td>
<td>PTE</td>
</tr>
<tr>
<td>Romanian leu</td>
<td>ROL</td>
</tr>
<tr>
<td>Russian ruble</td>
<td>RUR</td>
</tr>
<tr>
<td>Slovenian tolar</td>
<td>SIT</td>
</tr>
<tr>
<td>Spanish peseta</td>
<td>ESP</td>
</tr>
<tr>
<td>Swedish krona</td>
<td>SEK</td>
</tr>
<tr>
<td>Swiss franc</td>
<td>CHF</td>
</tr>
<tr>
<td>Turkish lira</td>
<td>TRL</td>
</tr>
</tbody>
</table>

The EUROCURR function converts a specific country's currency to an equivalent amount in another country's currency. It can also convert a specific country's currency to euros. EUROCURR uses the values in either the fixed currency conversion rate table or the changeable currency conversion rate table to convert currency.

If you are converting from one country's currency to euros, SAS divides by that from-currency-amount country's rate from one of the conversion rate tables. See “Example 1: Converting from Deutsche Marks to Euros” on page 941. If you are converting from euros to a country's currency, SAS multiplies by that from-currency-amount country's rate from one of the conversion rate tables. See “Example 2: Converting from Euros to Deutsche Marks” on page 942. If you are converting one country's currency to another country's currency, SAS first converts the from-currency-amount to euros. SAS stores the intermediate value in as much precision as your operating environment allows, and does not round the value. SAS then converts the amount in euros to an amount in the currency that you are converting to. See “Example 3: Converting from French Francs to Deutsche Marks” on page 942.

Examples:

Example 1: Converting from Deutsche Marks to Euros
The following example converts one Deutsche mark to an equivalent amount of euros.
data _null_;  
  amount=eurocurr(50,'dem','eur');  
  put amount=;  
run;  
The value in the SAS log is: amount=25.56459406.

Example 2: Converting from Euros to Deutsche Marks
The following example converts one euro to an equivalent amount of Deutsche marks.

data _null_;  
  amount=eurocurr(25,'eur','dem');  
  put amount=;  
run;  
The value in the SAS log is: amount=48.89575.

Example 3: Converting from French Francs to Deutsche Marks
The following example converts 50 French francs to an equivalent amount of Deutsche marks.

data _null_;  
  x=50;  
  amount=eurocurr(x,'frf','dem');  
  put amount=;  
run;  
The value in the SAS log is: amount=14.908218069.

Example 4: Converting Currency When One Variable Is Blank
The following example converts 50 euros to Deutsche marks.

data _null_;  
  x=50;  
  amount=eurocurr(x,' ','dem');  
  put amount=;  
run;  
The value in the SAS log is: amount=97.7915.
### Encodings and Their Aliases and Encoding Character Set Compatibility

This table lists common encoding methods and their corresponding encoding and alias names.

**Table A13.1  Encoding and Alias Names**

<table>
<thead>
<tr>
<th>Encoding Methods and Character Sets</th>
<th>Encoding Names in SAS</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicode</td>
<td>UTF-8</td>
<td>UTF-8</td>
</tr>
<tr>
<td>SBCS</td>
<td>WLATIN2</td>
<td>CP1250, WINDOWS-1250</td>
</tr>
<tr>
<td></td>
<td>WCYRILLIC</td>
<td>CP1251, WINDOWS-1251</td>
</tr>
<tr>
<td></td>
<td>WLATIN1</td>
<td>CP1252, WINDOWS-1252</td>
</tr>
<tr>
<td></td>
<td>WGREEK</td>
<td>CP1253, WINDOWS-1253</td>
</tr>
<tr>
<td></td>
<td>WTURKISH</td>
<td>CP1254, WINDOWS-1254</td>
</tr>
<tr>
<td></td>
<td>WHEBREW</td>
<td>CP1255, WINDOWS-1255</td>
</tr>
<tr>
<td></td>
<td>WARABIC</td>
<td>CP1256, WINDOWS-1256</td>
</tr>
<tr>
<td></td>
<td>WBALTIC</td>
<td>CP1257, WINDOWS-1257</td>
</tr>
<tr>
<td></td>
<td>WVIETNAMESE</td>
<td>CP1258, WINDOWS-1258</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>ISO8859_1, ISO8859-1</td>
</tr>
<tr>
<td>Encoding Methods and Character Sets</td>
<td>Encoding Names in SAS</td>
<td>Alias</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>LATIN2</td>
<td>ISO8859_2, ISO8859-2</td>
<td></td>
</tr>
<tr>
<td>LATIN3</td>
<td>ISO8859_3, ISO8859-3</td>
<td></td>
</tr>
<tr>
<td>CYRILLIC</td>
<td>ISO8859_5, ISO8859-5</td>
<td></td>
</tr>
<tr>
<td>ARABIC</td>
<td>ISO8859_6, ISO8859-6</td>
<td></td>
</tr>
<tr>
<td>GREEK</td>
<td>ISO8859_7, ISO8859-7</td>
<td></td>
</tr>
<tr>
<td>HEBREW</td>
<td>ISO8859_8, ISO8859-8</td>
<td></td>
</tr>
<tr>
<td>TURKISH</td>
<td>ISO8859_9, ISO8859-9</td>
<td></td>
</tr>
<tr>
<td>THAI</td>
<td>ISO8859_11, ISO8859-11</td>
<td></td>
</tr>
<tr>
<td>LATIN7</td>
<td>ISO8859_13, ISO8859-13</td>
<td></td>
</tr>
<tr>
<td>LATIN9</td>
<td>ISO8859_15, ISO8859-15</td>
<td></td>
</tr>
<tr>
<td>LATIN10</td>
<td>ISO8859_16, ISO8859-16</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-838</td>
<td>CP838</td>
<td></td>
</tr>
<tr>
<td>EBCDIC838</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-870</td>
<td>CP870</td>
<td></td>
</tr>
<tr>
<td>EBCDIC870</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-875</td>
<td>CP875</td>
<td></td>
</tr>
<tr>
<td>EBCDIC875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1047</td>
<td>CP1047</td>
<td></td>
</tr>
<tr>
<td>EBCDIC1047</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1122</td>
<td>CP1122</td>
<td></td>
</tr>
<tr>
<td>EBCDIC1122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1130, EBCDIC1130</td>
<td>CP1130</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1141, EBCDIC1141</td>
<td>CP1141</td>
<td></td>
</tr>
<tr>
<td>Encoding Methods and Character Sets</td>
<td>Encoding Names in SAS</td>
<td>Alias</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>OPEN_ED-1142</td>
<td>EBCDIC1142</td>
<td>CP1142</td>
</tr>
<tr>
<td>OPEN_ED-1143</td>
<td>EBCDIC1143</td>
<td>CP1143</td>
</tr>
<tr>
<td>OPEN_ED-1144</td>
<td>EBCDIC1144</td>
<td>CP1144</td>
</tr>
<tr>
<td>OPEN_ED-1145</td>
<td>EBCDIC1145</td>
<td>CP1145</td>
</tr>
<tr>
<td>OPEN_ED-1146</td>
<td>EBCDIC1146</td>
<td>CP1146</td>
</tr>
<tr>
<td>OPEN_ED-1147</td>
<td>EBCDIC1147</td>
<td>CP1147</td>
</tr>
<tr>
<td>OPEN_ED-1148</td>
<td>EBCDIC1148</td>
<td>CP1148</td>
</tr>
<tr>
<td>DBCS</td>
<td>SHIFT-JIS</td>
<td>MS-932, IBM-942, MACOS-1</td>
</tr>
<tr>
<td></td>
<td>MS-949</td>
<td>MACOS-3, EUC-KR</td>
</tr>
<tr>
<td></td>
<td>EUC-CN</td>
<td>MS-936, MACOS-25, DEC-CN, GBK, GB2312</td>
</tr>
<tr>
<td></td>
<td>EUC-TW</td>
<td>DEC-TW</td>
</tr>
<tr>
<td></td>
<td>MS-950</td>
<td>MACOS-2, BIG5</td>
</tr>
</tbody>
</table>

This table shows languages (locales) that support the same set of characters. For example, the Arabic locale in the first column has five compatible character sets: WARABIC, ARABIC, MSDOS720, PCOEM864, OPEN_ED-425. Encoding names used by SAS for the character sets are listed in the fourth column.
<table>
<thead>
<tr>
<th>Language</th>
<th>Language code (ISO 639)</th>
<th>Script group</th>
<th>Encoding names (in SAS)</th>
<th>Encoding names (in Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>ar</td>
<td>Arabic</td>
<td>WARABIC, ARABIC, MSDOS720, PCOEM864, OPEN_ED-425</td>
<td>CP1256, ISO8859_6, CP864, WINDOWS-1256</td>
</tr>
<tr>
<td>Bengali</td>
<td>bn</td>
<td>Bengali</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Persian</td>
<td>fa</td>
<td>Arabic</td>
<td>WARABIC, ARABIC, MSDOS720, PCOEM864, OPEN_ED-425</td>
<td>CP1256, ISO8859_6, CP864, WINDOWS-1256</td>
</tr>
<tr>
<td>Chinese (Simplified)</td>
<td>zh</td>
<td>Chinese (Simplified)</td>
<td>MS-936, EUC-CN, IBM-935</td>
<td>GBK, GBK, GB2312, CP935</td>
</tr>
<tr>
<td>Chinese (Traditional)</td>
<td>zh</td>
<td>Chinese (Traditional)</td>
<td>MS-950, EUC-TW, IBM-937</td>
<td>BIG5, BIG5, CP937</td>
</tr>
<tr>
<td>Belarusian</td>
<td>be</td>
<td>Cyrillic</td>
<td>WCYRILLIC, CYRILLIC, PCOEM866, OPEN_ED-1025</td>
<td>CP1251, ISO8859_5, CP866, CP1025, WINDOWS-1251</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>bg</td>
<td>Cyrillic</td>
<td>WCYRILLIC, CYRILLIC, PCOEM866, OPEN_ED-1025</td>
<td>CP1251, ISO8859_5, CP866, CP1025, WINDOWS-1251</td>
</tr>
<tr>
<td>Macedonian</td>
<td>mk</td>
<td>Cyrillic</td>
<td>WCYRILLIC, CYRILLIC, PCOEM866, OPEN_ED-1025</td>
<td>CP1251, ISO8859_5, CP866, CP1025, WINDOWS-1251</td>
</tr>
<tr>
<td>Russian</td>
<td>ru</td>
<td>Cyrillic</td>
<td>WCYRILLIC, CYRILLIC, PCOEM866, OPEN_ED-1025</td>
<td>CP1251, ISO8859_5, CP866, CP1025, WINDOWS-1251</td>
</tr>
<tr>
<td>Serbian</td>
<td>sr</td>
<td>Cyrillic</td>
<td>WCYRILLIC, CYRILLIC, PCOEM866, OPEN_ED-1025</td>
<td>CP1251, ISO8859_5, CP866, CP1025, WINDOWS-1251</td>
</tr>
<tr>
<td>Ukrainian</td>
<td>uk</td>
<td>Cyrillic</td>
<td>WCYRILLIC, CYRILLIC, PCOEM866, OPEN_ED-1025</td>
<td>CP1251, ISO8859_5, CP866, CP1025, WINDOWS-1251</td>
</tr>
<tr>
<td>Language</td>
<td>Language code (ISO 639)</td>
<td>Script group</td>
<td>Encoding names (in SAS)</td>
<td>Encoding names (in Java)</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
<td>--------------</td>
<td>------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Hindi</td>
<td>hi</td>
<td>Devanagari</td>
<td>PCISCI806, OPEN_ED-1137</td>
<td></td>
</tr>
<tr>
<td>Marathi</td>
<td>mr</td>
<td>Devanagari</td>
<td>PCISCI806, OPEN_ED-1137</td>
<td></td>
</tr>
<tr>
<td>Greek</td>
<td>el</td>
<td>Greek</td>
<td>WGREEK, GREEK, MSDOS737, PCOEM869, OPEN_ED-875</td>
<td>CP1253, ISO8859_7, CP869, CP875, WINDOWS-1253</td>
</tr>
<tr>
<td>Hebrew</td>
<td>he</td>
<td>Hebrew</td>
<td>WHEBREW, HEBREW, PCOEM862, OPEN_ED-424</td>
<td>CP1255, ISO8859_8, CP862, CP424, WINDOWS-1255</td>
</tr>
<tr>
<td>Afrikaans</td>
<td>af</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED- 1047</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1047, WINDOWS-1252</td>
</tr>
<tr>
<td>Catalan</td>
<td>ca</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED- 1145</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1145, WINDOWS-1252</td>
</tr>
<tr>
<td>Cornish</td>
<td>kw</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED- 1146</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1146, WINDOWS-1252</td>
</tr>
<tr>
<td>Danish</td>
<td>da</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED- 1142</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1142, WINDOWS-1252</td>
</tr>
<tr>
<td>Dutch</td>
<td>nl</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED- 1047</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1047, WINDOWS-1252</td>
</tr>
<tr>
<td>Language</td>
<td>Language code (ISO 639)</td>
<td>Script group</td>
<td>Encoding names (in SAS)</td>
<td>Encoding names (in Java)</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>---------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>English</td>
<td>en</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM437, OPEN_ED-1047</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP437, CP1047, WINDOWS-1252</td>
</tr>
<tr>
<td>Faroese</td>
<td>fo</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED-1142</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1142, WINDOWS-1252</td>
</tr>
<tr>
<td>Finnish</td>
<td>fi</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED-1143</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1143, WINDOWS-1252</td>
</tr>
<tr>
<td>French</td>
<td>fr</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED-1147</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1147, WINDOWS-1252</td>
</tr>
<tr>
<td>German</td>
<td>de</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED-1141</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1141, WINDOWS-1252</td>
</tr>
<tr>
<td>Greenlandic</td>
<td>kl</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED-1142</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1142, WINDOWS-1252</td>
</tr>
<tr>
<td>Icelandic</td>
<td>is</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED-1047</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1047, WINDOWS-1252</td>
</tr>
<tr>
<td>Indonesian</td>
<td>id</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED-1047</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1047, WINDOWS-1252</td>
</tr>
<tr>
<td>Language</td>
<td>Language code (ISO 639)</td>
<td>Script group</td>
<td>Encoding names (in SAS)</td>
<td>Encoding names (in Java)</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
<td>--------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Italian</td>
<td>it</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED- 1144</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1144, WINDOWS-1252</td>
</tr>
<tr>
<td>Malay</td>
<td>ms</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED- 1047</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1047, WINDOWS-1252</td>
</tr>
<tr>
<td>Manx</td>
<td>gv</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN8, PCOEM850, OPEN_ED- 1148</td>
<td>CP1252, ISO8859_14, CP850, CP1148, WINDOWS-1252</td>
</tr>
<tr>
<td>Norwegian</td>
<td>no</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED- 1142</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1142, WINDOWS-1252</td>
</tr>
<tr>
<td>Portuguese</td>
<td>pt</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED- 1047</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1047, WINDOWS-1252</td>
</tr>
<tr>
<td>Spanish</td>
<td>es</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED- 1145</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1145, WINDOWS-1252</td>
</tr>
<tr>
<td>Albanian</td>
<td>sq</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED- 870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Bosnian</td>
<td>bs</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED- 870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Croatian</td>
<td>hr</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED- 870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Language</td>
<td>Language code (ISO 639)</td>
<td>Script group</td>
<td>Encoding names (in SAS)</td>
<td>Encoding names (in Java)</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Czech</td>
<td>cs</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOE852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Hungarian</td>
<td>hu</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOE852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Polish</td>
<td>pl</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOE852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Romanian</td>
<td>ro</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOE852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Serbian (Latin)</td>
<td>sh</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOE852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Slovak</td>
<td>sk</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOE852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Slovenian</td>
<td>sl</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOE852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870, WINDOWS-1250</td>
</tr>
<tr>
<td>Maltese</td>
<td>mt</td>
<td>LATIN3</td>
<td>CP1254, ISO8859_9, CP857, CP1026</td>
<td>ISO8859_3</td>
</tr>
<tr>
<td>Turkish</td>
<td>tr</td>
<td>LATIN5</td>
<td>WTURKISH, TURKISH, PCOE857, OPEN_ED-1026</td>
<td>CP1254, ISO8859_9, CP857, CP1026, WINDOWS-1254</td>
</tr>
<tr>
<td>Language</td>
<td>Language code (ISO 639)</td>
<td>Script group</td>
<td>Encoding names (in SAS)</td>
<td>Encoding names (in Java)</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Estonian</td>
<td>et</td>
<td>LATIN6</td>
<td>WBALTIC, LATIN6, MSDOS775, PCOEM921, OPEN_ED-1112</td>
<td>CP1257, ISO8859_6, CP775, CP1112, WINDOWS-1257</td>
</tr>
<tr>
<td>Latvian</td>
<td>lv</td>
<td>LATIN6</td>
<td>WBALTIC, LATIN6, MSDOS775, PCOEM921, OPEN_ED-1112</td>
<td>CP1257, ISO8859_6, CP775, CP1112, WINDOWS-1257</td>
</tr>
<tr>
<td>Lithuanian</td>
<td>lt</td>
<td>LATIN6</td>
<td>WBALTIC, LATIN6, MSDOS775, PCOEM921, OPEN_ED-1112</td>
<td>CP1257, ISO8859_6, CP775, CP1112, WINDOWS-1257</td>
</tr>
<tr>
<td>Tamil</td>
<td>ta</td>
<td>Tamil</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Telugu</td>
<td>te</td>
<td>Telegu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thai</td>
<td>th</td>
<td>Thai</td>
<td>PCOEM874, THAI</td>
<td>CP874, TIS620</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>vi</td>
<td>Vietnamese</td>
<td>WVIENTNAMESE, OPEN_ED-1130</td>
<td>CP1258, CP1130, WINDOWS-1258</td>
</tr>
</tbody>
</table>
Appendix 3

Time Zone IDs and Time Zone Names

| Area: Africa |  | 953 |
| Area: America–North, Central, and South |  | 956 |
| Area: Antarctica |  | 966 |
| Area: Arctic |  | 966 |
| Area: Asia |  | 967 |
| Area: Atlantic |  | 972 |
| Area: Australia |  | 973 |
| Area: Miscellaneous |  | 975 |
| Area: Europe |  | 981 |
| Area: Pacific |  | 988 |

Area: Africa

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa/Abidjan</td>
<td>GMT</td>
<td>CI</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Accra</td>
<td>GMT</td>
<td>GH</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Algiers</td>
<td>CET</td>
<td>DZ</td>
<td>Central European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Bamako</td>
<td>GMT</td>
<td>ML</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Bangui</td>
<td>WAT</td>
<td>CF</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Banjul</td>
<td>GMT</td>
<td>GM</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Bissau</td>
<td>GMT</td>
<td>GW</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>----------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Africa/Brazzaville</td>
<td>WAT</td>
<td>CG</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Casablanca</td>
<td>+01</td>
<td>MA</td>
<td>Western European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Casablanca</td>
<td>+00</td>
<td>MA</td>
<td>Western European Summer Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Ceuta</td>
<td>CET</td>
<td>ES</td>
<td>Central European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Ceuta</td>
<td>CEST</td>
<td>ES</td>
<td>Central European Summer Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Conakry</td>
<td>GMT</td>
<td>GN</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Dakar</td>
<td>GMT</td>
<td>SN</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Douala</td>
<td>WAT</td>
<td>CM</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/El_Aaiun</td>
<td>+01</td>
<td>EH</td>
<td>Western European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/El_Aaiun</td>
<td>+00</td>
<td>EH</td>
<td>Western European Summer Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Freetown</td>
<td>GMT</td>
<td>SL</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Lome</td>
<td>GMT</td>
<td>TG</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Monrovia</td>
<td>GMT</td>
<td>LR</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Nouakchott</td>
<td>GMT</td>
<td>MR</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Ouagadougou</td>
<td>GMT</td>
<td>BF</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Sao_Tome</td>
<td>GMT</td>
<td>ST</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Timbuktu</td>
<td>GMT</td>
<td>CI</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Tunis</td>
<td>CET</td>
<td>TN</td>
<td>Central European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Kinshasa</td>
<td>WAT</td>
<td>CD</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Lagos</td>
<td>WAT</td>
<td>NG</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Libreville</td>
<td>WAT</td>
<td>GA</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Luanda</td>
<td>WAT</td>
<td>AO</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Malabo</td>
<td>WAT</td>
<td>GQ</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Africa/Ndjamena</td>
<td>WAT</td>
<td>TD</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Niamey</td>
<td>WAT</td>
<td>NE</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Porto-Novo</td>
<td>WAT</td>
<td>BJ</td>
<td>West Africa Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Windhoek</td>
<td>CAT</td>
<td>NA</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Blantyre</td>
<td>CAT</td>
<td>MW</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Bujumbura</td>
<td>CAT</td>
<td>BI</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Gaborone</td>
<td>CAT</td>
<td>BW</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Harare</td>
<td>CAT</td>
<td>ZW</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Kigali</td>
<td>CAT</td>
<td>RW</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Lubumbashi</td>
<td>CAT</td>
<td>CD</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Lusaka</td>
<td>CAT</td>
<td>ZM</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Maputo</td>
<td>CAT</td>
<td>MZ</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Cairo</td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Tripoli</td>
<td>EET</td>
<td>LY</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Johannesburg</td>
<td>SAST</td>
<td>ZA</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Maseru</td>
<td>SAST</td>
<td>LS</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Mbabane</td>
<td>SAST</td>
<td>SZ</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Addis_Ababa</td>
<td>EAT</td>
<td>ET</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Asmara</td>
<td>EAT</td>
<td>ET</td>
<td>East Africa Time</td>
<td>3.00</td>
</tr>
<tr>
<td>Africa/Asmera</td>
<td>EAT</td>
<td>ER</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Dar_es_Salaam</td>
<td>EAT</td>
<td>TZ</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Djibouti</td>
<td>EAT</td>
<td>DJ</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Juba</td>
<td>EAT</td>
<td>SS</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Kampala</td>
<td>EAT</td>
<td>UG</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Khartoum</td>
<td>CAT</td>
<td>SD</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Mogadishu</td>
<td>EAT</td>
<td>SO</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
</tbody>
</table>
### Africa

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa/Nairobi</td>
<td>EAT</td>
<td>KE</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
</tbody>
</table>

### Area: America—North, Central, and South

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>America/Adak</td>
<td>HST</td>
<td>US</td>
<td>Hawaii–Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>America/Adak</td>
<td>HDT</td>
<td>US</td>
<td>Hawaii–Aleutian Daylight Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Anchorage</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Anchorage</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Anguilla</td>
<td>AST</td>
<td>AI</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Antigua</td>
<td>AST</td>
<td>AG</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Araguaina</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Buenos_Aires</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Catamarca</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/ComodRivadavia</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Cordoba</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Jujuy</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/La_Rioja</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Mendoza</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Rio_Gallegos</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Salta</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/San_Juan</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Regio</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Argentina/San_Luis</td>
<td>-03</td>
<td>AR</td>
<td>Western Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Tucuman</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Argentina/Ushuaia</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Aruba</td>
<td>AST</td>
<td>AW</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Asuncion</td>
<td>-04</td>
<td>PY</td>
<td>Paraguay Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Asuncion</td>
<td>-03</td>
<td>PY</td>
<td>Paraguay Summer Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Atikokan</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Atka</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>America/Atka</td>
<td>HDT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Bahia</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Bahia_Banderas</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Bahia_Banderas</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Barbados</td>
<td>AST</td>
<td>BB</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Belem</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Belize</td>
<td>CST</td>
<td>BZ</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Blanc-Sablon</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Boa_Vista</td>
<td>-04</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Bogota</td>
<td>-05</td>
<td>CO</td>
<td>Colombia Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Boise</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Boise</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Buenos_Aires</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Cambridge_Bay</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Cambridge_Bay</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Campo_Grande</td>
<td>-04</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Campo_Grande</td>
<td>-03</td>
<td>BR</td>
<td>Amazon Summer Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Cancun</td>
<td>EST</td>
<td>MX</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Caracas</td>
<td>-04 VE</td>
<td></td>
<td>Venezuela Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Catamarca</td>
<td>-03 AR</td>
<td></td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Cayenne</td>
<td>-03 GF</td>
<td></td>
<td>French Guiana Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Cayman</td>
<td>EST KY</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Chicago</td>
<td>CST US</td>
<td></td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Chicago</td>
<td>CDT US</td>
<td></td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Chihuahua</td>
<td>MST MX</td>
<td></td>
<td>Mexican Pacific Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Chihuahua</td>
<td>MDT MX</td>
<td></td>
<td>Mexican Pacific Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Coral_Harbour</td>
<td>EST CA</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Cordoba</td>
<td>-03 AR</td>
<td></td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Costa_Rica</td>
<td>CST CR</td>
<td></td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Creston</td>
<td>MST CA</td>
<td></td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Cuiaba</td>
<td>-04 BR</td>
<td></td>
<td>Amazon Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Cuiaba</td>
<td>-03 BR</td>
<td></td>
<td>Amazon Summer Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Curacao</td>
<td>AST CW</td>
<td></td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Danmarkshavn</td>
<td>GMT GL</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>America/Dawson</td>
<td>PST CA</td>
<td></td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Dawson</td>
<td>PDT CA</td>
<td></td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Dawson_Creek</td>
<td>MST CA</td>
<td></td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Denver</td>
<td>MST US</td>
<td></td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Denver</td>
<td>MDT US</td>
<td></td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Detroit</td>
<td>EST US</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Detroit</td>
<td>EDT US</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Dominica</td>
<td>AST DM</td>
<td></td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Edmonton</td>
<td>MST CA</td>
<td></td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Edmonton</td>
<td>MDT CA</td>
<td></td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Eirunepe</td>
<td>-05 BR</td>
<td></td>
<td>Acre Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>America/El_Salvador</td>
<td>CST</td>
<td>SV</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Ensenada</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Ensenada</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Fort_Nelson</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Fort_Wayne</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Fort_Wayne</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Fortaleza</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Glace_Bay</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Glace_Bay</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Godthab</td>
<td>-03</td>
<td>GL</td>
<td>West Greenland Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Godthab</td>
<td>-02</td>
<td>GL</td>
<td>West Greenland Summer Time</td>
<td>-02:00</td>
</tr>
<tr>
<td>America/Goose_Bay</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Goose_Bay</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EDT</td>
<td>TC</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EST</td>
<td>TC</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Grenada</td>
<td>AST</td>
<td>GD</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Guadeloupe</td>
<td>AST</td>
<td>GP</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Guatemala</td>
<td>CST</td>
<td>GT</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Guayaquil</td>
<td>-05</td>
<td>EC</td>
<td>Ecuador Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Guyana</td>
<td>-04</td>
<td>GY</td>
<td>Guyana Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Halifax</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Halifax</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Havana</td>
<td>CST</td>
<td>CU</td>
<td>Cuba Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Havana</td>
<td>CDT</td>
<td>CU</td>
<td>Cuba Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Hermosillo</td>
<td>MST</td>
<td>MX</td>
<td>Mexican Pacific Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Indiana/Vincennes</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indiana/Vincennes</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Indiana/Winamac</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indiana/Winamac</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Indianapolis</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Indianapolis</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Inuvik</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Inuvik</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Iqaluit</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Iqaluit</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Jamaica</td>
<td>EST</td>
<td>JM</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Jujuy</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Juneau</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Juneau</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Kentucky/Louisville</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>America/Kentucky/Louisville</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Kentucky/Monticello</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Kentucky/Monticello</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Knox_IN</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Knox_IN</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Kralendijk</td>
<td>AST</td>
<td>BQ</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/La_Paz</td>
<td>-04</td>
<td>BO</td>
<td>Bolivia Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Lima</td>
<td>-05</td>
<td>PE</td>
<td>Peru Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Los_Angeles</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Los_Angeles</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Louisville</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Louisville</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Lower_Princes</td>
<td>AST</td>
<td>SX</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Maceio</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Managua</td>
<td>CST</td>
<td>NI</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Manaus</td>
<td>-04</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Marigot</td>
<td>AST</td>
<td>MF</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Martinique</td>
<td>AST</td>
<td>MQ</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Matamoros</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Matamoros</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Mazatlan</td>
<td>MST</td>
<td>MX</td>
<td>Mexican Pacific Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Mazatlan</td>
<td>MDT</td>
<td>MX</td>
<td>Mexican Pacific Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Mendoza</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Menominee</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Menominee</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Merida</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>America/Merida</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Metlakatla</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Metlakatla</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Mexico_City</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Mexico_City</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Miquelon</td>
<td>-03 PM</td>
<td>PM</td>
<td>St Pierre and Miquelon Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Miquelon</td>
<td>-02 PM</td>
<td>PM</td>
<td>St Pierre and Miquelon Daylight Time</td>
<td>-02:00</td>
</tr>
<tr>
<td>America/Moncton</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Moncton</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Monterrey</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Monterrey</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Montevideo</td>
<td>-03</td>
<td>UY</td>
<td>Uruguay Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Montreal</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Montreal</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Montserrat</td>
<td>AST</td>
<td>MS</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Nassau</td>
<td>EST</td>
<td>BS</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Nassau</td>
<td>EDT</td>
<td>BS</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/New_York</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/New_York</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Nipigon</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Nipigon</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Nome</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Nome</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Noronha</td>
<td>-02</td>
<td>BR</td>
<td>Fernando de Noronha Standard Time</td>
<td>-02:00</td>
</tr>
<tr>
<td>America/North_Dakota/Beulah</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>America/North_Dakota/Beulah</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/North_Dakota/Center</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/North_Dakota/Center</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/North_Dakota/New_Salem</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/North_Dakota/New_Salem</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Ojinaga</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Ojinaga</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Paramaribo</td>
<td>-03</td>
<td>SR</td>
<td>Suriname Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Phoenix</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Port-au-Prince</td>
<td>EST</td>
<td>PA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Port-au-Prince</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Port_of_Spain</td>
<td>AST</td>
<td>TT</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Porto_Acre</td>
<td>-05</td>
<td>BR</td>
<td>Acre Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Porto_Velho</td>
<td>-04</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Puerto_Rico</td>
<td>AST</td>
<td>PR</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Punta_Arenas</td>
<td>-03</td>
<td>CL</td>
<td>Chile Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Rainy_River</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Rainy_River</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Rankin_Inlet</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Rankin_Inlet</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Recife</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Regina</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Resolute</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Resolute</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Rio_Branco</td>
<td>-05</td>
<td>BR</td>
<td>Acre Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Rosario</td>
<td>-03</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Santa_Isabel</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Santa_Isabel</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Santarem</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Santiago</td>
<td>-04</td>
<td>CL</td>
<td>Chile Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Santiago</td>
<td>-03</td>
<td>CL</td>
<td>Chile Summer Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Santo_Domingo</td>
<td>AST</td>
<td>DO</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Sao_Paulo</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Sao_Paulo</td>
<td>-02</td>
<td>BR</td>
<td>Brasilia Summer Time</td>
<td>-02:00</td>
</tr>
<tr>
<td>America/Scoresbysund</td>
<td>-01</td>
<td>GL</td>
<td>East Greenland Standard Time</td>
<td>-01:00</td>
</tr>
<tr>
<td>America/Scoresbysund</td>
<td>+00</td>
<td>GL</td>
<td>East Greenland Summer Time</td>
<td>00:00</td>
</tr>
<tr>
<td>America/Shiprock</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Shiprock</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Sitka</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Sitka</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/St_Barthelemy</td>
<td>AST</td>
<td>BL</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/St_Johns</td>
<td>NST</td>
<td>CA</td>
<td>Newfoundland Standard Time</td>
<td>-03:30</td>
</tr>
<tr>
<td>America/St_Johns</td>
<td>NDT</td>
<td>CA</td>
<td>Newfoundland Daylight Time</td>
<td>-02:30</td>
</tr>
<tr>
<td>America/St_Kitts</td>
<td>AST</td>
<td>KN</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/St_Lucia</td>
<td>AST</td>
<td>LC</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/St_Thomas</td>
<td>AST</td>
<td>VI</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/St_Vincent</td>
<td>AST</td>
<td>VC</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Swift_Current</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>--------</td>
<td>------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Tegucigalpa</td>
<td>CST</td>
<td>HN</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Thule</td>
<td>AST</td>
<td>GL</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Thule</td>
<td>ADT</td>
<td>GL</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>America/Thunder_Bay</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Thunder_Bay</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Tijuana</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Tijuana</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Toronto</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Toronto</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Tortola</td>
<td>AST</td>
<td>VG</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Vancouver</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Vancouver</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Virgin</td>
<td>AST</td>
<td>AI</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>America/Whitehorse</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Whitehorse</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Winnipeg</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>America/Winnipeg</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>America/Yakutat</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>America/Yakutat</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>America/Yellowknife</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>America/Yellowknife</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
</tbody>
</table>
## Area: Antarctica

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antarctica/Casey</td>
<td>+08</td>
<td>AQ</td>
<td>Australian Western Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Antarctica/Davis</td>
<td>+07</td>
<td>AQ</td>
<td>Davis Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Antarctica/DumontDUrville</td>
<td>+10</td>
<td>AQ</td>
<td>Dumont-d'Urville Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Antarctica/Macquarie</td>
<td>+11</td>
<td>AU</td>
<td>Macquarie Island Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Antarctica/Mawson</td>
<td>+05</td>
<td>AQ</td>
<td>Mawson Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Antarctica/McMurdo</td>
<td>NZST</td>
<td>AQ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Antarctica/McMurdo</td>
<td>NZDT</td>
<td>AQ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Antarctica/Palmer</td>
<td>-03</td>
<td>AQ</td>
<td>Chile Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>Antarctica/Palmer</td>
<td>CLST</td>
<td>AQ</td>
<td>Chile Summer Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Antarctica/Rothera</td>
<td>-03</td>
<td>AQ</td>
<td>Rothera Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Antarctica/South_Pole</td>
<td>NZST</td>
<td>AQ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Antarctica/South_Pole</td>
<td>NZDT</td>
<td>AQ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Antarctica/Syowa</td>
<td>+03</td>
<td>AQ</td>
<td>Syowa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Antarctica/Troll</td>
<td>+00</td>
<td>AQ</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Antarctica/Troll</td>
<td>+02</td>
<td>AQ</td>
<td>Greenwich Mean Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Antarctica/Vostok</td>
<td>+06</td>
<td>AQ</td>
<td>Vostok Time</td>
<td>06:00</td>
</tr>
</tbody>
</table>

## Area: Arctic

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arctic/Longyearbyen</td>
<td>CET</td>
<td>SJ</td>
<td>Central European Standard Time</td>
<td>01:00</td>
</tr>
</tbody>
</table>
### Time Zone Information

<table>
<thead>
<tr>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central European Summer Time</td>
<td>02:00</td>
</tr>
</tbody>
</table>

### Area: Asia

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia/Aden</td>
<td>+03</td>
<td>YE</td>
<td>Arabian Standard Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Almaty</td>
<td>+06</td>
<td>KZ</td>
<td>East Kazakhstan Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Amman</td>
<td>EET</td>
<td>JO</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Asia/Amman</td>
<td>EEST</td>
<td>JO</td>
<td>Eastern European Summer Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Anadyr</td>
<td>+12</td>
<td>RU</td>
<td>Anadyr Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Asia/Aqtau</td>
<td>+05</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Aqtobe</td>
<td>+05</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Ashgabat</td>
<td>+05</td>
<td>TM</td>
<td>Turkmenistan Standard Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Ashkhabad</td>
<td>+05</td>
<td>TM</td>
<td>Turkmenistan Standard Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Atyrau</td>
<td>+05</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Baghdad</td>
<td>+03</td>
<td>IQ</td>
<td>Arabian Standard Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Bahrain</td>
<td>+03</td>
<td>BH</td>
<td>Arabian Standard Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Baku</td>
<td>+04</td>
<td>AZ</td>
<td>Azerbaijan Standard Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Asia/Baku</td>
<td>+05</td>
<td>AZ</td>
<td>Azerbaijan Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Bangkok</td>
<td>+07</td>
<td>TH</td>
<td>Indochina Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Barnaul</td>
<td>+07</td>
<td>RU</td>
<td>Krasnoyarsk Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Beijing</td>
<td>CST</td>
<td>CN</td>
<td>China Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Beirut</td>
<td>EET</td>
<td>LB</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>-------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Asia/Beirut</td>
<td>EEST</td>
<td>LB</td>
<td>Eastern European Summer Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Bishkek</td>
<td>+06</td>
<td>KG</td>
<td>Kyrgyzstan Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Brunei</td>
<td>+08</td>
<td>BN</td>
<td>Brunei Darussalam Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Calcutta</td>
<td>IST</td>
<td>IN</td>
<td>India Standard Time</td>
<td>05:30</td>
</tr>
<tr>
<td>Asia/Chita</td>
<td>+09</td>
<td>RU</td>
<td>Yakutsk Standard Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Choibalsan</td>
<td>+08</td>
<td>MN</td>
<td>Choibalsan Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Choibalsan</td>
<td>CHOST</td>
<td>MN</td>
<td>Choibalsan Summer Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Chongqing</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Chungking</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Colombo</td>
<td>+0530</td>
<td>LK</td>
<td>India Standard Time</td>
<td>05:30</td>
</tr>
<tr>
<td>Asia/Dacca</td>
<td>+06</td>
<td>BD</td>
<td>Bangladesh Standard Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Damascus</td>
<td>EET</td>
<td>SY</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Asia/Damascus</td>
<td>EEST</td>
<td>SY</td>
<td>Eastern European Summer Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Dhaka</td>
<td>+06</td>
<td>BD</td>
<td>Bangladesh Standard Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Dili</td>
<td>+09</td>
<td>TL</td>
<td>East Timor Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Dubai</td>
<td>+04</td>
<td>AE</td>
<td>Gulf Standard Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Asia/Dushanbe</td>
<td>+05</td>
<td>TJ</td>
<td>Tajikistan Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Famagusta</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Asia/Famagusta</td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Gaza</td>
<td>EET</td>
<td>PS</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Asia/Gaza</td>
<td>EEST</td>
<td>PS</td>
<td>Eastern European Summer Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Harbin</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Hebron</td>
<td>EET</td>
<td>PS</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Time Zone Name</td>
<td>Time Zone Offset</td>
<td>Region</td>
<td>Time Zone Description</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>------------------</td>
<td>--------</td>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td>Asia/Hebron</td>
<td>+03</td>
<td>PS</td>
<td>Eastern European Summer Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Ho_Chi Minh</td>
<td>+07</td>
<td>VN</td>
<td>Indochina Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Hong_Kong</td>
<td></td>
<td>HK</td>
<td>Hong Kong Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Hovd</td>
<td>+07</td>
<td>MN</td>
<td>Hovd Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Hovd</td>
<td>HOVST</td>
<td>MN</td>
<td>Hovd Summer Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Irkutsk</td>
<td>+08</td>
<td>RU</td>
<td>Irkutsk Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Istanbul</td>
<td>+03</td>
<td>TR</td>
<td>Eastern European Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Jakarta</td>
<td>WIB</td>
<td>ID</td>
<td>Western Indonesia Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Jayapura</td>
<td>WIT</td>
<td>ID</td>
<td>Eastern Indonesia Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Jerusalem</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Jerusalem</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Kabul</td>
<td>+0430</td>
<td>AF</td>
<td>Afghanistan Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Kamchatka</td>
<td>+12</td>
<td>RU</td>
<td>Petropavlovsk-Kamchatski Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Karachi</td>
<td>PKT</td>
<td>PK</td>
<td>Pakistan Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Kashgar</td>
<td>+06</td>
<td>CN</td>
<td>China Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Katmandu</td>
<td>+0545</td>
<td>NP</td>
<td>Nepal Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Khandyga</td>
<td>+09</td>
<td>RU</td>
<td>Yakutsk Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Kolkata</td>
<td>IST</td>
<td>IN</td>
<td>India Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Krasnoyarsk</td>
<td>+07</td>
<td>RU</td>
<td>Krasnoyarsk Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Kuala_Lumpur</td>
<td>+08</td>
<td>MY</td>
<td>Malaysia Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Kuching</td>
<td>+08</td>
<td>MY</td>
<td>Malaysia Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Kuwait</td>
<td>+03</td>
<td>KW</td>
<td>Arabian Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Macao</td>
<td>CST</td>
<td>MO</td>
<td>China Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Macao</td>
<td>CST</td>
<td>MO</td>
<td>China Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Magadan</td>
<td>+11</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td></td>
</tr>
<tr>
<td>Asia/Makassar</td>
<td>WITA</td>
<td>ID</td>
<td>Central Indonesia Time</td>
<td></td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Regio n</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>---------</td>
<td>----------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Asia/Manila</td>
<td>PST</td>
<td>PH</td>
<td>Philippine Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Muscat</td>
<td>+04</td>
<td>OM</td>
<td>Gulf Standard Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Asia/Nicosia</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Asia/Nicosia</td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Novokuznetsk</td>
<td>+07</td>
<td>RU</td>
<td>Krasnoyarsk Standard Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Novosibirsk</td>
<td>+07</td>
<td>RU</td>
<td>Novosibirsk Standard Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Omsk</td>
<td>+06</td>
<td>RU</td>
<td>Omsk Standard Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Qostanay</td>
<td>+06</td>
<td>KZ</td>
<td>East Kazakhstan Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Oral</td>
<td>+05</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Osaka</td>
<td>JST</td>
<td>JP</td>
<td>Japan Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Phnom_Penh</td>
<td>+07</td>
<td>KH</td>
<td>Indochina Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Pontianak</td>
<td>WIB</td>
<td>ID</td>
<td>Western Indonesia Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Pyongyang</td>
<td>KST</td>
<td>KP</td>
<td>Korean Standard Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Qatar</td>
<td>+03</td>
<td>QA</td>
<td>Arabian Standard Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Qyzylorda</td>
<td>+05</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Rangoon</td>
<td>+0630</td>
<td>MM</td>
<td>Myanmar Time</td>
<td>06:30</td>
</tr>
<tr>
<td>Asia/Riyadh</td>
<td>+03</td>
<td>SA</td>
<td>Arabian Standard Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Saigon</td>
<td>+07</td>
<td>VN</td>
<td>Indochina Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Sakhalin</td>
<td>+11</td>
<td>RU</td>
<td>Sakhalin Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Samarkand</td>
<td>+05</td>
<td>UZ</td>
<td>Uzbekistan Standard Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Sapporo</td>
<td>JST</td>
<td>JP</td>
<td>Japan Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Seoul</td>
<td>KST</td>
<td>KR</td>
<td>Korean Standard Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Shanghai</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Singapore</td>
<td>+08</td>
<td>SG</td>
<td>Singapore Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Srednekolymsk</td>
<td>+11</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Taipei</td>
<td>CST</td>
<td>TW</td>
<td>Taipei Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Asia/Tashkent</td>
<td>+05</td>
<td>UZ</td>
<td>Uzbekistan Standard Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Tbilisi</td>
<td>+04</td>
<td>GE</td>
<td>Georgia Standard Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Asia/Tehran</td>
<td>+0330</td>
<td>IR</td>
<td>Iran Standard Time</td>
<td>03:30</td>
</tr>
<tr>
<td>Asia/Tehran</td>
<td>+0430</td>
<td>IR</td>
<td>Iran Daylight Time</td>
<td>04:30</td>
</tr>
<tr>
<td>Asia/Tel_Aviv</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Asia/Tel_Aviv</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Asia/Thimbu</td>
<td>+06</td>
<td>BT</td>
<td>Bhutan Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Thimphu</td>
<td>BTT</td>
<td>BT</td>
<td>Bhutan Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Asia/Tokyo</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Tomsk</td>
<td>+07</td>
<td>RU</td>
<td>Omsk Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Ujung_Pandang</td>
<td>WITA</td>
<td>ID</td>
<td>Central Indonesia Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Ulan_Bator</td>
<td>+08</td>
<td>MN</td>
<td>Ulaanbaatar Bator Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Urumqi</td>
<td>XJT</td>
<td>CN</td>
<td>China Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Asia/Ust-Nera</td>
<td>+10</td>
<td>RU</td>
<td>Vladivostok Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Asia/Vientiane</td>
<td>+07</td>
<td>LA</td>
<td>Indochina Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Asia/Vladivostok</td>
<td>+10</td>
<td>RU</td>
<td>Vladivostok Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Asia/Yakutsk</td>
<td>+09</td>
<td>RU</td>
<td>Yakutsk Standard Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Asia/Yanqon</td>
<td>+0630</td>
<td>MM</td>
<td></td>
<td>06:30</td>
</tr>
<tr>
<td>Asia/Yekaterinburg</td>
<td>+05</td>
<td>RU</td>
<td>Yekaterinburg Standard Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Asia/Yerevan</td>
<td>+04</td>
<td>AM</td>
<td>Armenia Standard Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Asia/Yerevan</td>
<td>+05</td>
<td>AM</td>
<td>Armenia Time</td>
<td>05:00</td>
</tr>
</tbody>
</table>
### Time Zone Information

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlantic/Azores</td>
<td>AZOT</td>
<td>PT</td>
<td>Azores Time</td>
<td>-01:00</td>
</tr>
<tr>
<td>Atlantic/Azores</td>
<td>+00</td>
<td>PT</td>
<td>Azores Summer Standard Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Atlantic/Bermuda</td>
<td>AST</td>
<td>BM</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>Atlantic/Bermuda</td>
<td>ADT</td>
<td>BM</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Atlantic/Canary</td>
<td>WET</td>
<td>ES</td>
<td>Western European Standard Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Atlantic/Canary</td>
<td>WEST</td>
<td>ES</td>
<td>Western European Summer Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Atlantic/Cape_Verde</td>
<td>-01</td>
<td>CV</td>
<td>Cape Verde Standard Time</td>
<td>-01:00</td>
</tr>
<tr>
<td>Atlantic/Faeroe</td>
<td>WET</td>
<td>FO</td>
<td>Western European Standard Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Atlantic/Faeroe</td>
<td>WEST</td>
<td>FO</td>
<td>Western European Summer Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Atlantic/Faroe</td>
<td>WET</td>
<td>FO</td>
<td>Western European Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Atlantic/Faroe</td>
<td>WEST</td>
<td>FO</td>
<td>Western European Summer Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CET</td>
<td>SJ</td>
<td>Central European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CEST</td>
<td>SJ</td>
<td>Central European Summer Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Atlantic/Reykjavik</td>
<td>GMT</td>
<td>IS</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Atlantic/South_Georgia</td>
<td>-02</td>
<td>GS</td>
<td>South Georgia Time</td>
<td>-02:00</td>
</tr>
<tr>
<td>Atlantic/St_Helena</td>
<td>GMT</td>
<td>SH</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Atlantic/Stanley</td>
<td>-03</td>
<td>FK</td>
<td>Falkland Islands Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Australia/ACT</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/ACT</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>ACST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>09:30</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>ACDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Brisbane</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>ACST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>09:30</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>ACDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Darwin</td>
<td>ACST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>09:30</td>
</tr>
<tr>
<td>Australia/Eucla</td>
<td>+0845</td>
<td>AU</td>
<td>Australian Central Western Standard Time</td>
<td>08:45</td>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>+1030</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>+11</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Lindeman</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>+1030</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>+11</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/North</td>
<td>ACST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>09:30</td>
</tr>
<tr>
<td>Australia/Perth</td>
<td>AWST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Australia/Queensland</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/South</td>
<td>ACST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>09:30</td>
</tr>
<tr>
<td>Australia/South</td>
<td>ACDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>AEST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>AEDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Australia/West</td>
<td>AWST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>ACST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>09:30</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>ACDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Brazil/Acre</td>
<td>-05</td>
<td>BR</td>
<td>Acre Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Brazil/DeNoronha</td>
<td>-02</td>
<td>BR</td>
<td>Fernando de Noronha Standard Time</td>
<td>-02:00</td>
</tr>
<tr>
<td>Brazil/East</td>
<td>-03</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Brazil/East</td>
<td>-02</td>
<td>BR</td>
<td>Brasilia Summer Time</td>
<td>-02:00</td>
</tr>
<tr>
<td>Brazil/West</td>
<td>-04</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>CST6CDT</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>CST6CDT</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Canada/Atlantic</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>Canada/Atlantic</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Canada/Central</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Canada/Central</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Canada/East-Saskatchewan</td>
<td>CST</td>
<td>CA</td>
<td>Central Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Canada/Eastern</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Canada/Eastern</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>Canada/Mountain</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>Canada/Mountain</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Canada/Newfoundland</td>
<td>NST</td>
<td>CA</td>
<td>Newfoundland Standard Time</td>
<td>-03:30</td>
</tr>
<tr>
<td>Canada/Newfoundland</td>
<td>NDT</td>
<td>CA</td>
<td>Newfoundland Daylight Time</td>
<td>-02:30</td>
</tr>
<tr>
<td>Canada/Pacific</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>Canada/Pacific</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>Canada/Saskatchewan</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Canada/Yukon</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>Canada/Yukon</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>Chile/Continental</td>
<td>-04</td>
<td>CL</td>
<td>Chile Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>Chile/Continental</td>
<td>-03</td>
<td>CL</td>
<td>Chile Summer Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>Chile/EasterIsland</td>
<td>-06</td>
<td>CL</td>
<td>Easter Island Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Chile/EasterIsland</td>
<td>-05</td>
<td>CL</td>
<td>Easter Island Summer Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Cuba</td>
<td>CST</td>
<td>CU</td>
<td>Cuba Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Cuba</td>
<td>CDT</td>
<td>CU</td>
<td>Cuba Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>EST5EDT</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>EST5EDT</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>Egypt</td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Eire</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Eire</td>
<td>IST</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Etc/GMT</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/GMT+0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/GMT+1</td>
<td>GMT-01</td>
<td></td>
<td>1 hours behind GMT</td>
<td>-01:00</td>
</tr>
<tr>
<td>Etc/GMT+2</td>
<td>GMT-02</td>
<td></td>
<td>2 hours behind GMT</td>
<td>-02:00</td>
</tr>
<tr>
<td>Etc/GMT+3</td>
<td>GMT-03</td>
<td></td>
<td>3 hours behind GMT</td>
<td>-03:00</td>
</tr>
<tr>
<td>Etc/GMT+4</td>
<td>GMT-04</td>
<td></td>
<td>4 hours behind GMT</td>
<td>-04:00</td>
</tr>
<tr>
<td>Etc/GMT+5</td>
<td>GMT-05</td>
<td></td>
<td>5 hours behind GMT</td>
<td>-05:00</td>
</tr>
<tr>
<td>Etc/GMT+6</td>
<td>GMT-06</td>
<td></td>
<td>6 hours behind GMT</td>
<td>-06:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Etc/GMT+7</td>
<td>GMT-07</td>
<td></td>
<td>7 hours behind GMT</td>
<td>-07:00</td>
</tr>
<tr>
<td>Etc/GMT+8</td>
<td>GMT-08</td>
<td></td>
<td>8 hours behind GMT</td>
<td>-08:00</td>
</tr>
<tr>
<td>Etc/GMT+9</td>
<td>GMT-09</td>
<td></td>
<td>9 hours behind GMT</td>
<td>-09:00</td>
</tr>
<tr>
<td>Etc/GMT+10</td>
<td>GMT-10</td>
<td></td>
<td>10 hours behind GMT</td>
<td>-10:00</td>
</tr>
<tr>
<td>Etc/GMT+11</td>
<td>GMT-11</td>
<td></td>
<td>11 hours behind GMT</td>
<td>-11:00</td>
</tr>
<tr>
<td>Etc/GMT+12</td>
<td>GMT-12</td>
<td></td>
<td>12 hours behind GMT</td>
<td>-12:00</td>
</tr>
<tr>
<td>Etc/GMT-0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/GMT-1</td>
<td>GMT+01</td>
<td></td>
<td>1 hour ahead of GMT</td>
<td>01:00</td>
</tr>
<tr>
<td>Etc/GMT-2</td>
<td>GMT+02</td>
<td></td>
<td>2 hours ahead of GMT</td>
<td>02:00</td>
</tr>
<tr>
<td>Etc/GMT-3</td>
<td>GMT+03</td>
<td></td>
<td>3 hours ahead of GMT</td>
<td>03:00</td>
</tr>
<tr>
<td>Etc/GMT-4</td>
<td>GMT+04</td>
<td></td>
<td>4 hours ahead of GMT</td>
<td>04:00</td>
</tr>
<tr>
<td>Etc/GMT-5</td>
<td>GMT+05</td>
<td></td>
<td>5 hours ahead of GMT</td>
<td>05:00</td>
</tr>
<tr>
<td>Etc/GMT-6</td>
<td>GMT+06</td>
<td></td>
<td>6 hours ahead of GMT</td>
<td>06:00</td>
</tr>
<tr>
<td>Etc/GMT-7</td>
<td>GMT+07</td>
<td></td>
<td>7 hours ahead of GMT</td>
<td>07:00</td>
</tr>
<tr>
<td>Etc/GMT-8</td>
<td>GMT+08</td>
<td></td>
<td>8 hours ahead of GMT</td>
<td>08:00</td>
</tr>
<tr>
<td>Etc/GMT-9</td>
<td>GMT+09</td>
<td></td>
<td>9 hours ahead of GMT</td>
<td>09:00</td>
</tr>
<tr>
<td>Etc/GMT-10</td>
<td>GMT+10</td>
<td></td>
<td>10 hours ahead of GMT</td>
<td>10:00</td>
</tr>
<tr>
<td>Etc/GMT-11</td>
<td>GMT+11</td>
<td></td>
<td>11 hours ahead of GMT</td>
<td>11:00</td>
</tr>
<tr>
<td>Etc/GMT-12</td>
<td>GMT+12</td>
<td></td>
<td>12 hours ahead of GMT</td>
<td>12:00</td>
</tr>
<tr>
<td>Etc/GMT-13</td>
<td>GMT+13</td>
<td></td>
<td>13 hours ahead of GMT</td>
<td>13:00</td>
</tr>
<tr>
<td>Etc/GMT-14</td>
<td>GMT+14</td>
<td></td>
<td>14 hours ahead of GMT</td>
<td>14:00</td>
</tr>
<tr>
<td>Etc/GMT0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/Greenwich</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/UCT</td>
<td>UCT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/UTC</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/Universal</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Etc/Zulu</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>GB</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>GB</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>01:00</td>
</tr>
<tr>
<td>GB-Eire</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>GB-Eire</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>01:00</td>
</tr>
<tr>
<td>GMT</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>GMT+0</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>GMT-0</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>GMT0</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Greenwich</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Hongkong</td>
<td>HKT</td>
<td>HK</td>
<td>Hong Kong Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>Iceland</td>
<td>GMT</td>
<td>IS</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Indian/Antananarivo</td>
<td>EAT</td>
<td>MG</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Indian/Chagos</td>
<td>+06</td>
<td>IO</td>
<td>Indian Ocean Time</td>
<td>06:00</td>
</tr>
<tr>
<td>Indian/Christmas</td>
<td>+07</td>
<td>CX</td>
<td>Christmas Island Time</td>
<td>07:00</td>
</tr>
<tr>
<td>Indian/Cocos</td>
<td>+0630</td>
<td>CC</td>
<td>Cocos Islands Time</td>
<td>06:30</td>
</tr>
<tr>
<td>Indian/Comoro</td>
<td>EAT</td>
<td>KM</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Indian/Kerguelen</td>
<td>+05</td>
<td>TF</td>
<td>French Southern and Antarctic Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Indian/Mahe</td>
<td>+04</td>
<td>SC</td>
<td>Seychelles Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Indian/Maldives</td>
<td>+05</td>
<td>MV</td>
<td>Maldives Time</td>
<td>05:00</td>
</tr>
<tr>
<td>Indian/Mauritius</td>
<td>+04</td>
<td>MU</td>
<td>Mauritius Standard Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Indian/Mayotte</td>
<td>EAT</td>
<td>YT</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Indian/Reunion</td>
<td>+04</td>
<td>RE</td>
<td>Reunion Time</td>
<td>04:00</td>
</tr>
<tr>
<td>Iran</td>
<td>+0330</td>
<td>IR</td>
<td>Iran Standard Time</td>
<td>03:30</td>
</tr>
<tr>
<td>Iran</td>
<td>+0430</td>
<td>IR</td>
<td>Iran Daylight Time</td>
<td>04:30</td>
</tr>
<tr>
<td>Israel</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Israel</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Jamaica</td>
<td>EST</td>
<td>JM</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Japan</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Kwajalein</td>
<td>+12</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Libya</td>
<td>EET</td>
<td>LY</td>
<td>Eastern European Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>PST</td>
<td>MX</td>
<td>Mexican Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>PDT</td>
<td>MX</td>
<td>Mexican Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MST</td>
<td>MX</td>
<td>Mexican Pacific Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MDT</td>
<td>MX</td>
<td>Mexican Pacific Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Mexico/General</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Mexico/General</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>NZ</td>
<td>NZST</td>
<td>AQ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>NZ</td>
<td>NZDT</td>
<td>AQ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>NZ-CHAT</td>
<td>+1245</td>
<td>NZ</td>
<td>Chatham Standard Time</td>
<td>12:45</td>
</tr>
<tr>
<td>NZ-CHAT</td>
<td>+1345</td>
<td>NZ</td>
<td>Chatham Daylight Time</td>
<td>13:45</td>
</tr>
<tr>
<td>Navajo</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>Navajo</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>PRC</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>PST8PDT</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>PST8PDT</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>Poland</td>
<td>CET</td>
<td>PL</td>
<td>Central European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Poland</td>
<td>CEST</td>
<td>PL</td>
<td>Central European Summer Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Portugal</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>--------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Portugal</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>01:00</td>
</tr>
<tr>
<td>ROC</td>
<td>CST</td>
<td>TW</td>
<td>Taipei Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>ROK</td>
<td>KST</td>
<td>KR</td>
<td>Korean Standard Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Singapore</td>
<td>+08</td>
<td>SG</td>
<td>Singapore Standard Time</td>
<td>08:00</td>
</tr>
<tr>
<td>SystemV/AST4ADT</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>SystemV/AST4ADT</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-03:00</td>
</tr>
<tr>
<td>SystemV/CST6CDT</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>SystemV/CST6CDT</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>SystemV/EST5EDT</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>SystemV/EST5EDT</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>SystemV/MST7MDT</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>SystemV/MST7MDT</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>SystemV/PST8PDT</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>SystemV/PST8PDT</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>SystemV/YST9YDT</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>SystemV/YST9YDT</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>Turkey</td>
<td>+03</td>
<td>TR</td>
<td>Eastern European Time</td>
<td>03:00</td>
</tr>
<tr>
<td>UCT</td>
<td>UCT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>US/Alaska</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>US/Alaska</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>US/Aleutian</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>US/Aleutian</td>
<td>HDT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>US/Arizona</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>US/Central</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>US/Central</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>US/East-Indiana</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
</tbody>
</table>
### Time Zone Information

<table>
<thead>
<tr>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>US/East-Indiana</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>US/Eastern</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>US/Eastern</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>US/Hawaii</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-04:00</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-07:00</td>
</tr>
<tr>
<td>US/Samoa</td>
<td>UM</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>UTC</td>
<td>UM</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Universal</td>
<td>UM</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>W-SU</td>
<td>RU</td>
<td>Moscow Standard Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Zulu</td>
<td>UTC</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
</tbody>
</table>

### Area: Europe

<table>
<thead>
<tr>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe/Amsterdam</td>
<td>NL</td>
<td>Central European Standard Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Europe/Amsterdam</td>
<td>CEST</td>
<td>NL</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CET</td>
<td>AD</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CEST</td>
<td>AD</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Astrakhan</td>
<td>+04</td>
<td>RU</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Europe/Athens</td>
<td>EET</td>
<td>GR</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Athens</td>
<td>EEST</td>
<td>GR</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CET</td>
<td>RS</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CEST</td>
<td>RS</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CET</td>
<td>SK</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CEST</td>
<td>SK</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CET</td>
<td>BE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CEST</td>
<td>BE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EET</td>
<td>RO</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EEST</td>
<td>RO</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CET</td>
<td>HU</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CEST</td>
<td>HU</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CET</td>
<td>DK</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CEST</td>
<td>DK</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>IST</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CET</td>
<td>GI</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CEST</td>
<td>GI</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>GMT</td>
<td>GG</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>BST</td>
<td>GG</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Helsinki</td>
<td>EET</td>
<td>FI</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Helsinki</td>
<td>EEST</td>
<td>FI</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>GMT</td>
<td>IM</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>BST</td>
<td>IM</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Istanbul</td>
<td>+03</td>
<td>TR</td>
<td>Eastern European Time</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>GMT</td>
<td>JE</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>BST</td>
<td>JE</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Kaliningrad</td>
<td>EET</td>
<td>RU</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Kirov</td>
<td>+03</td>
<td>RU</td>
<td>Moscow Time</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CET</td>
<td>SI</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CEST</td>
<td>SI</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/London</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/London</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CET</td>
<td>LU</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CEST</td>
<td>LU</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CET</td>
<td>ES</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CEST</td>
<td>ES</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CET</td>
<td>MT</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CEST</td>
<td>MT</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EET</td>
<td>AX</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EEST</td>
<td>AX</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Minsk</td>
<td>+03</td>
<td>BY</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CET</td>
<td>MC</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CEST</td>
<td>MC</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Moscow</td>
<td>MSK</td>
<td>RU</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Europe/Nicosia</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td></td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Oslo</td>
<td>CET</td>
<td>NO</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td>NO</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Paris</td>
<td>CET</td>
<td>FR</td>
<td>Central European Time</td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td>FR</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Podgorica</td>
<td>CET</td>
<td>ME</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td>ME</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Prague</td>
<td>CET</td>
<td>CZ</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td>CZ</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Riga</td>
<td>EET</td>
<td>LV</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td></td>
<td>EEST</td>
<td>LV</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Rome</td>
<td>CET</td>
<td>IT</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td>IT</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Samara</td>
<td>+04</td>
<td>RU</td>
<td>Samara Standard Time</td>
</tr>
<tr>
<td>Europe/San_Marino</td>
<td>CET</td>
<td>SM</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td>SM</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CET</td>
<td>BA</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CEST</td>
<td>BA</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Saratov</td>
<td>+04</td>
<td>RU</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Europe/Simferopol</td>
<td>MSK</td>
<td>UA</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CET</td>
<td>MK</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CEST</td>
<td>MK</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EET</td>
<td>BG</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EEST</td>
<td>BG</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CET</td>
<td>SE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CEST</td>
<td>SE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EET</td>
<td>EE</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EEST</td>
<td>EE</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Tirane</td>
<td>CET</td>
<td>AL</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Tirane</td>
<td>CEST</td>
<td>AL</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Ulyanovsk</td>
<td>+04</td>
<td>RU</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CET</td>
<td>LI</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CEST</td>
<td>LI</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CET</td>
<td>VA</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CEST</td>
<td>VA</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CET</td>
<td>AT</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CEST</td>
<td>AT</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EET</td>
<td>LT</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EEST</td>
<td>LT</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Volgograd</td>
<td>+04</td>
<td>RU</td>
<td>Volgograd Standard Time</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CET</td>
<td>PL</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CEST</td>
<td>PL</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CET</td>
<td>HR</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CEST</td>
<td>HR</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CET</td>
<td>CH</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CEST</td>
<td>CH</td>
<td>Central European Summer Time</td>
</tr>
</tbody>
</table>
## Area: Pacific

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pacific/Apia</td>
<td>+13</td>
<td>WS</td>
<td>Apia Standard Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Apia</td>
<td>+14</td>
<td>WS</td>
<td>Apia Daylight Time</td>
<td>14:00</td>
</tr>
<tr>
<td>Pacific/Auckland</td>
<td>NZST</td>
<td>NZ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Auckland</td>
<td>NZDT</td>
<td>NZ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Bougainville</td>
<td>+11</td>
<td>PG</td>
<td>Papua New Guinea Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Chatham</td>
<td>+1245</td>
<td>NZ</td>
<td>Chatham Standard Time</td>
<td>12:45</td>
</tr>
<tr>
<td>Pacific/Chatham</td>
<td>+1345</td>
<td>NZ</td>
<td>Chatham Daylight Time</td>
<td>13:45</td>
</tr>
<tr>
<td>Pacific/Chuuk</td>
<td>+10</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Easter</td>
<td>-06</td>
<td>CL</td>
<td>Easter Island Standard Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Pacific/Easter</td>
<td>-05</td>
<td>CL</td>
<td>Easter Island Summer Time</td>
<td>-05:00</td>
</tr>
<tr>
<td>Pacific/Efate</td>
<td>+11</td>
<td>VU</td>
<td>Vanuatu Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Enderbury</td>
<td>+13</td>
<td>KI</td>
<td>Phoenix Islands Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Fakaofo</td>
<td>+13</td>
<td>TK</td>
<td>Tokelau Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Fiji</td>
<td>+12</td>
<td>FJ</td>
<td>Fiji Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Fiji</td>
<td>+13</td>
<td>FJ</td>
<td>Fiji Summer Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Funafuti</td>
<td>+12</td>
<td>TV</td>
<td>Tuvalu Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Galapagos</td>
<td>-06</td>
<td>EC</td>
<td>Galapagos Time</td>
<td>-06:00</td>
</tr>
<tr>
<td>Pacific/Gambier</td>
<td>-09</td>
<td>PF</td>
<td>Gambier Time</td>
<td>-09:00</td>
</tr>
<tr>
<td>Pacific/Guadalcanal</td>
<td>+11</td>
<td>SB</td>
<td>Solomon Islands Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Guam</td>
<td>CHST</td>
<td>GU</td>
<td>Chamorro Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Honolulu</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Johnston</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Kiritimati</td>
<td>+14</td>
<td>KI</td>
<td>Line Islands Time</td>
<td>14:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------</td>
<td>--------</td>
<td>---------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Pacific/Kosrae</td>
<td>+11</td>
<td>FM</td>
<td>Kosrae Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Kwajalein</td>
<td>+12</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Majuro</td>
<td>+12</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Marquesas</td>
<td>-0930</td>
<td>PF</td>
<td>Marquesas Time</td>
<td>-09:30</td>
</tr>
<tr>
<td>Pacific/Midway</td>
<td>SST</td>
<td>UM</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Nauru</td>
<td>+12</td>
<td>NR</td>
<td>Nauru Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Niue</td>
<td>-11</td>
<td>NU</td>
<td>Niue Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Norfolk</td>
<td>+11</td>
<td>NF</td>
<td>Norfolk Island Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Noumea</td>
<td>+11</td>
<td>NC</td>
<td>New Caledonia Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Pago_Pago</td>
<td>SST</td>
<td>AS</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Palau</td>
<td>+09</td>
<td>PW</td>
<td>Palau Time</td>
<td>09:00</td>
</tr>
<tr>
<td>Pacific/Pitcairn</td>
<td>-08</td>
<td>PN</td>
<td>Pitcairn Time</td>
<td>-08:00</td>
</tr>
<tr>
<td>Pacific/Pohnpei</td>
<td>PONT</td>
<td>FM</td>
<td>Ponape Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Ponape</td>
<td>+11</td>
<td>FM</td>
<td>Ponape Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Port_Moresby</td>
<td>+10</td>
<td>PG</td>
<td>Papua New Guinea Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Rarotonga</td>
<td>-10</td>
<td>CK</td>
<td>Cook Islands Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Saipan</td>
<td>CHST</td>
<td>MP</td>
<td>Chamorro Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Samoa</td>
<td>SST</td>
<td>UM</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Tahiti</td>
<td>-10</td>
<td>PF</td>
<td>Tahiti Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Tarawa</td>
<td>+12</td>
<td>KI</td>
<td>Gilbert Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Tongatapu</td>
<td>+13</td>
<td>TO</td>
<td>Tonga Standard Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Tongatapu</td>
<td>+14</td>
<td>TO</td>
<td>Tonga Summer Time</td>
<td>14:00</td>
</tr>
<tr>
<td>Pacific/Truk</td>
<td>+10</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Wake</td>
<td>+12</td>
<td>UM</td>
<td>Wake Island Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Wallis</td>
<td>+12</td>
<td>WF</td>
<td>Wallis and Futuna Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Yap</td>
<td>+10</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
</tbody>
</table>
%COPY_TO_NEW.Encoding

Macro

Dictionary

%COPY_TO_NEW.Encoding Macro Function

Creates a new version of a data set with a specified encoding.

Category: Encoding

Syntax

%COPY_TO_NEW.Encoding(from_dsname, to_dsname, new_encoding)

Arguments

from_dsname
specifies the original data set name.

to_dsname
specifies the new data set name.

new_encoding
specifies the encoding of the new data set.

Details

Basics

The %COPY_TO_NEW.Encoding macro creates a new version of a data set with a specified encoding. If the data set contains character variables whose values need larger lengths when transcoded to the specified encoding, then the DATA step creates a data set with the proper lengths. If the %COPY_TO_NEW.Encoding macro is not used, the copy might fail because truncation of non-blanks is not
allowed. For example, the following code, with a euc-cn encoding, fails because the variable x needs three bytes to hold the transcoded UTF-8 representation:

```plaintext
data temp;
x='d6d0'x;
run;
data temp(encoding=utf8);
set temp;
run;
```

The %COPY_TO_NEW_ENCODING macro produces this type of code:

```plaintext
data temp(encoding=utf8);
length x $3;
set temp;
run;
```

%COPY_TO_NEW_ENCODING Macro Code

Here is the code for the %COPY_TO_NEW_ENCODING macro:

```plaintext
%macro copy_to_new_encoding(from_dsname,to_dsname,new_encoding);
%global orig_encoding;

%let prefix=goobly;

filename lngtstmt temp;
data _null_;   
  file lngtstmt;   
  put ' ';        
run;

filename kcvtused temp;
data _null_;   
  file kcvtused;   
  put ' ';        
run;

data temp2;
  x=1;
run;

%global sql_libname sql_memname;
data _null_;   
  length libname memname $256;   
  memname=scan("&from_dsname",-1,'.');   
  libname=ifc(index("&from_dsname","."),scan("&from_dsname",1,'.'),"WORK");   
  call symputx('sql_libname',upcase(libname));   
  call symputx('sql_memname',upcase(memname));   
run;
proc sql;
  create table temp as select * from dictionary.tables   
    where libname="&sql_libname." and memname="&sql_memname.";
quit;
data _null_;   
  set temp;
```
call symputx('orig_encoding',scan(encoding,1,' '));
run;

proc contents data=&from_dsname out=temp(keep=name type length npos)
noprint;
run;

proc sort data=temp;
  by name;
run;

%global nchars revise;
%let revise=0;
data _null_
  set temp end=eof;
  retain nchars 0;
  nchars + (type=2);
  if eof;
    call symputx('nchars',nchars);
run;

%if &nchars %then %do;
data temp2(keep=&prefix._name &prefix._length
  rename=(&prefix._name=NAME));
set &from_dsname(encoding=binary) end=&prefix._eof;
retain &prefix._revise 0;
array &prefix._charlens{&nchars} _temporary_
array &prefix._charvars _character_
if _n_=1 then do over &prefix._charvars;
  &prefix._charlens{_i_}= -vlength(&prefix._charvars);
end;
do over &prefix._charvars;
  &prefix._l = lengthc(kcvt(trim(&prefix._charvars),"&orig_encoding.","&new_encoding."));
if &prefix._l > abs(&prefix._charlens{_i_}) then do;
  &prefix._charlens{_i_} = &prefix._l;
  &prefix._revise = 1;
end;
end;
if &prefix._eof and &prefix._revise;
call symputx('revise',1);
length &prefix._name $32 &prefix._length 8;
do over &prefix._charvars;
  if &prefix._charlens{_i_} > 0 then do;
    &prefix._name = vname(&prefix._charvars);
    &prefix._length = &prefix._charlens{_i_};
    output temp2;
  end;
end;
run;

%if &revise %then %do;

%COPY_TO_NEW_ENCODING Macro Function 993
994 Chapter 26 / %COPY_TO_NEW_ENCODING Macro

proc sort data=temp2;
by name;
run;
data temp; merge temp temp2(in=revised);
by name;
if revised then length=&prefix._length;
need_kcvt = revised;
run;
proc sort;
by npos;
run;
data _null_;
set temp;
file lngtstmt mod;
length nlit $512 stmt $1024;
nlit = nliteral(name);
len = cats(ifc(type=2,'$',' '),length);
stmt = catx(' ','length',nlit,len,';');
put stmt;
if need_kcvt;
stmt = trim(nlit)||' = kcvt('||trim(nlit)||",""&orig_encoding
."",""&new_encoding."");";
put stmt;
run;
%end;
%end;
data &to_dsname(encoding=&new_encoding);
%include lngtstmt/source2;
set &from_dsname(encoding=binary);
%include kcvtused/source2;
run;
filename lngtstmt clear;
filename kcvtused clear;
proc delete data=temp temp2;
run;
%mend copy_to_new_encoding;

%COPY_TO_NEW_ENCODING Macro with Explanations
Here is the code for the %COPY_TO_NEW_ENCODING macro with explanations:
Statements

Details

%macro copy_to_new_encoding(from_dsname,to_dsname,new_encoding);

Invoke the
%COPY_TO_NEW_ENCODING
macro.

%global orig_encoding;


$let prefix=goobly;

Use a prefix to ensure unique variable names.

filename lngtstmt temp;
data _null_
   file lngtstmt;
   put ' ';
run;

Create a LENGTH statement file with a blank line.

filename kcvtused temp;
data_null_
   file kcvtused;
   put ' ';
run;

Create a KCVT usage file with a blank.

data temp2;
   x=1;
run;

Create a temp2 data set that might be replaced before being deleted.

%global sql_libname sql_memname;
data_null_
   length libname memname $256;
   memname=scan("&from_dsname",-1,'.');
   libname=ifc(index("&from_dsname".','),scan("&from_dsname",1,'.'),'WORK');
   call symputx('sql_libname',upcase(libname));
   call symputx('sql_memname',upcase(memname));
run;

proc sql;
   create table temp as select * from dictionary.tables
      where libname="&sql_libname." and memname="&sql_memname.";
quit;

data _null_; set temp;
call symputx('orig_encoding',scan(encoding,1,' '));
run;

Get the encoding for the data set. Although this does appear in a PROC CONTENTS output, the label 'Encoding' might be in another language. So the safe way to do this is to get the encoding column for the data set via dictionary.tables, making sure to use the WHERE clause so that only the one data set is retrieved.

proc contents data=&from_dsname out=temp
   (keep=name type length npos) noprint;
run;

Sort by name for later updates.

%global nchars revise;
%let revise=0;
data_null_; set temp end=eof;
   retain nchars 0;
   nchars + (type=2);
   if eof;
      call symputx('nchars',nchars);
run;

Macro variable revise is set to 1 if a revision is needed. Macro variable nchars indicates the number of character variables found.
%if &nchars %then %do;
  data temp2(keep=&prefix._name &prefix._length
    rename=(&prefix._name=NAME));
  set &from_dsname(encoding=binary) end=&prefix._eof;
  retain &prefix._revise 0;
  array &prefix._charlens{&nchars} _temporary_;
  array &prefix._charvars _character_;
  if _n_=1 then do over &prefix._charvars;
    &prefix._charlens{_i_} = -vlength(&prefix._charvars);
  end;
  do over &prefix._charvars;
    &prefix._l = lengthc(kcvt(trim(&prefix._charvars),
      "&orig_encoding.","&new_encoding.");
    if &prefix._l > abs(&prefix._charlens{_i_}) then do;
      &prefix._charlens{_i_} = &prefix._l;
    end;
    &prefix._revise = 1;
  end;
  if &prefix._eof and &prefix._revise;
    call symputx('revise',1);
    length &prefix._name $32 &prefix._length 8;
    do over &prefix._charvars;
      if &prefix._charlens{_i_} > 0 then do;
        &prefix._name = vname(&prefix._charvars);
        &prefix._length = &prefix._charlens{_i_};
        output temp2;
      end;
    end;
  run;
%if &revise %then %do;
  %if &revise %then %do;
  If any lengths are revised, then create a LENGTH statement.
  proc sort data=temp2; by name; run;
  data temp; merge temp temp2(in=revised); by name;
    if revised then length=&prefix._length;
    need_kcvt = revised;
  run;
  proc sort; by npos;
  run;
  Merge the revised lengths.
  Sort backward to npos to maintain the original order.
  proc sort data=temp2; by name; run;
  data temp; merge temp temp2(in=revised); by name;
    if revised then length=&prefix._length;
    need_kcvt = revised;
  run;
  proc sort; by npos;
  run;
  Merge the revised lengths.
  Sort backward to npos to maintain the original order.
  proc sort data=temp2; by name; run;
  data temp; merge temp temp2(in=revised); by name;
    if revised then length=&prefix._length;
    need_kcvt = revised;
  run;
  proc sort; by npos;
  run;
  Merge the revised lengths.
  Sort backward to npos to maintain the original order.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; set temp;</td>
<td>Generate a LENGTH statement for all variables in order.</td>
</tr>
<tr>
<td>file lngtstmt mod;</td>
<td></td>
</tr>
<tr>
<td>length nlit $512 stmt $1024;</td>
<td></td>
</tr>
<tr>
<td>nlit = nliteral(name);</td>
<td></td>
</tr>
<tr>
<td>len = cats(ifc(type=2,‘$‘,' '),length);</td>
<td></td>
</tr>
<tr>
<td>stmt = catx(' ','length',nlit,len,';');</td>
<td></td>
</tr>
<tr>
<td>put stmt;</td>
<td></td>
</tr>
<tr>
<td>if need_kcvt;</td>
<td></td>
</tr>
<tr>
<td>stmt = trim(nlit)</td>
<td></td>
</tr>
<tr>
<td>“”,”&amp;new_encoding.”<em>);</em>;</td>
<td></td>
</tr>
<tr>
<td>put stmt;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>$end;                      $end;</td>
<td></td>
</tr>
</tbody>
</table>

| data &to dsname(encoding=&new_encoding);                                  | Create the new data set with the original or revised lengths.         |
|                           &include lngtstmt/source2;                          |                                                                        |
|                           &set &from dsname(encoding=binary);               |                                                                        |
|                           &include kcvtused/source2;                         |                                                                        |
|                           run;                                          |                                                                        |

| filename lngtstmt clear;                                                 | Cleanup                                                                |
| filename kcvtused clear;                                                 |                                                                        |
| proc delete data=temp temp2;                                             |                                                                        |
| run;                                                                     |                                                                        |
| $mend copy_to_new_encoding;                                              |                                                                        |