# Contents


## PART 1  NLS Concepts  1

**Chapter 1 • National Language Support** ............................................................ 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 ....................................... 13  
Code Point Discrepancies among EBCDIC Encodings .............................. 13  
Collating Sequence ............................................................................... 14  
Determining the Encoding of a SAS Data Set ....................................... 16  
Encoding Behavior in a SAS Session .................................................... 17  

**Chapter 4 • Transcoding for NLS** .......................................................... 19  
Overview to Transcoding .................................................................... 19  
Common Reasons for Transcoding ....................................................... 20  
SAS Options That Transcode SAS Data ................................................. 20  
Transcoding Considerations .................................................................. 21  
Compatible and Incompatible Encodings .............................................. 21  
Preventing Transcoding ....................................................................... 22  
Avoiding Character Data Truncation By Using the CVP Engine .......... 23  

**Chapter 5 • Time Zones** ................................................................. 25  
Overview of SAS Time Zone IDs ............................................................ 25  
Specifying Time Zones in SAS ............................................................ 25  

## PART 2  Autocall Macros for NLS  33

**Chapter 6 • Dictionary of Autocall Macros for NLS** .............................. 35  
Autocall Macro Entries by Category ....................................................... 35  
Dictionary ......................................................................................... 35  

## PART 3  Data Set Options for NLS  39
<table>
<thead>
<tr>
<th>Chapter 7 • Dictionary of Data Set Options for NLS</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Options by Category</td>
<td>41</td>
</tr>
<tr>
<td>Dictionary</td>
<td>41</td>
</tr>
</tbody>
</table>

**PART 4 Formats for NLS** 45

<table>
<thead>
<tr>
<th>Chapter 8 • Overview to NLS Formats</th>
<th>47</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Date and Datetime Formats</td>
<td>47</td>
</tr>
<tr>
<td>Currency Representation</td>
<td>53</td>
</tr>
<tr>
<td>Exceptions for Date and Time Default Widths</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 9 • Dictionary of Formats for NLS</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categories of NLS Formats</td>
<td>66</td>
</tr>
<tr>
<td>Dictionary</td>
<td>82</td>
</tr>
</tbody>
</table>

**PART 5 Functions for NLS** 277

<table>
<thead>
<tr>
<th>Chapter 10 • Internationalization Compatibility for SAS String Functions</th>
<th>279</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internationalization Compatibility for SAS String Functions</td>
<td>279</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 11 • VARCHAR Support in Functions</th>
<th>299</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR Data Type in String Functions</td>
<td>299</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 12 • Dictionary of NLS Call Routines and Functions</th>
<th>303</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions by Category</td>
<td>304</td>
</tr>
<tr>
<td>Dictionary</td>
<td>310</td>
</tr>
</tbody>
</table>

**PART 6 Informats for NLS** 431

<table>
<thead>
<tr>
<th>Chapter 13 • Dictionary of Informats for NLS</th>
<th>433</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informats by Category</td>
<td>435</td>
</tr>
<tr>
<td>Dictionary</td>
<td>442</td>
</tr>
</tbody>
</table>

**PART 7 Macro Functions for NLS** 559

<table>
<thead>
<tr>
<th>Chapter 14 • Dictionary of Macro Functions for NLS</th>
<th>561</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro Functions by Category</td>
<td>561</td>
</tr>
<tr>
<td>Dictionary</td>
<td>562</td>
</tr>
</tbody>
</table>

**PART 8 System Options for NLS** 571

<table>
<thead>
<tr>
<th>Chapter 15 • Dictionary of System Options for NLS</th>
<th>573</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Option Entries by Category</td>
<td>573</td>
</tr>
<tr>
<td>Dictionary</td>
<td>574</td>
</tr>
</tbody>
</table>

Overview

One CALL Routine and seven functions are new. An appendix with encoding tables has been added. Several functions have been updated.

New CALL Routines, Functions and Appendixes

CALL Routines and Functions

These CALL routines and functions are new:

- **CALL KSCANX** (p. 316)
  - Returns the position and length of the $n$th word from a character string.

- **KCOUNTC** (p. 329)
  - Counts the number of words in a character string.

- **KCOUNTW** (p. 331)
  - Counts the number of words in a character string.

- **KCOUNTX** (p. 334)
  - Counts the number of words in a character string.

- **KFIND** (p. 337)
  - Searches for a specific substring of characters within a character string.

- **KFINDC** (p. 339)
  - Searches a string for any character in a list of characters.

- **KFINDW** (p. 342)
  - Returns the character position of a word in a string or the number of the word in a string.

- **KSCANX** (p. 365)
  - Selects a specified word from a character expression.

Appendix

This appendix is new:
Updated CALL Routines and Functions

These functions and CALL routines have been updated:

**KLEFT** (p. 350)

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

**KSTRIP** (p. 370)

Information about data normalization has been added.
Part 1

NLS Concepts

Chapter 1
National Language Support ........................................... 3

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

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

Chapter 4
Transcoding for NLS ................................................. 19

Chapter 5
Time Zones .............................................................. 25
Chapter 1
National Language Support

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 uppercasing and lowercasing 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 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
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 language 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 279.
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 a 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 the 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 it 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, which 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, a 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, while 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 the 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
- MAPECBDICTOASCII
- 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 629.

The English_UnitedStates value for LOCALE= implicitly sets the following options to the specified default values at SAS invocation:

- DATESTYLE=MDY
- DFLANG=English
- ENCODING=UTF-8
- PAPERSIZE=Letter
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

**How the 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 Viya 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 startup. 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 startup. 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 585. 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 580.

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 582.
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 the SAS configuration. If **ODSLANGCHG=ON**, all messages are in English because of the LSWLANG setting. For more information, see “LOGLANGENG System Option” on page 581.

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.

UTF-8 is the only SAS session encoding supported by SAS Viya.

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 (the 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.

### Difference between Encoding and Transcoding

Encoding establishes the default working environment for your SAS session. UTF-8 is the only SAS session encoding supported by SAS Viya.

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. For SAS Viya, which supports only UTF-8, 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. For conceptual information, see Chapter 4, “Transcoding for NLS,” on page 19.

### 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 13.

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.

UTF-8 is the only SAS session encoding supported by SAS Viya.

Common encoding methods are listed here:

**Unicode** is a common and popular character set.

**Unicode** provides up to 110,116 character combinations. Unicode can accommodate basically all of the world's languages. Unicode is a coded character set. HTTP and HTML protocols are based on Unicode. The following UTF encodings are part 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 only SAS session encoding supported by SAS Viya.
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 13.

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.

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>Latin1</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 Standard</td>
<td>Name of Encoding</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>ISO 8859-4</td>
<td>Baltic</td>
<td>Northern European</td>
</tr>
<tr>
<td>ISO 8859-5</td>
<td>Cyrillic</td>
<td>Slavic languages</td>
</tr>
<tr>
<td>ISO 8859-6</td>
<td>Arabic</td>
<td>Arabic</td>
</tr>
<tr>
<td>ISO 8859-7</td>
<td>Greek</td>
<td>Modern Greek</td>
</tr>
<tr>
<td>ISO 8859-8</td>
<td>Hebrew</td>
<td>Hebrew and Yiddish</td>
</tr>
<tr>
<td>ISO 8859-9</td>
<td>Turkish</td>
<td>Turkish</td>
</tr>
<tr>
<td>ISO 8859-10</td>
<td>Latin 6</td>
<td>Nordic (Inuit, Sámi, Icelandic)</td>
</tr>
<tr>
<td>ISO 8859-11</td>
<td>Latin/Thai</td>
<td>Thai</td>
</tr>
<tr>
<td>ISO 8859-13</td>
<td>Latin 7</td>
<td>Baltic Rim</td>
</tr>
<tr>
<td>ISO 8859-14</td>
<td>Latin 8</td>
<td>Celtic</td>
</tr>
<tr>
<td>ISO 8859-15</td>
<td>Latin 9</td>
<td>Western European</td>
</tr>
<tr>
<td>ISO-8859-16</td>
<td>Latin 10</td>
<td>Central and Eastern European</td>
</tr>
</tbody>
</table>

Also, a number of encoding standards have been developed for East Asian languages. Some of these are listed in the following table.

**Table 3.2 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</td>
</tr>
</tbody>
</table>

There are other encodings in the standards for EBCDIC and Windows that support different languages and locales. See “Encodings and their Character Set Compatibility and Aliases” on page 655 for common encodings and their aliases.
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 to facilitate the free flow of goods and services between nations and to advocate for the exchange of intellectual, scientific, and technological information. 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 develops and promotes the Unicode standard, which provides a unique number for every character.

Table 3.3  Variant Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>1025</th>
<th>1047</th>
<th>870</th>
<th>838</th>
<th>1142</th>
<th>1141</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>4F</td>
<td>BB</td>
<td>5A</td>
<td>4F</td>
<td>4F</td>
</tr>
<tr>
<td>#</td>
<td>7B</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>
</tr>
<tr>
<td>$</td>
<td>5B</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>
</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></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>
</tr>
<tr>
<td>[</td>
<td></td>
<td>AD</td>
<td>49</td>
<td>4A</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>
</tr>
<tr>
<td>]</td>
<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>
</tr>
<tr>
<td>^</td>
<td></td>
<td>5F</td>
<td>69</td>
<td>5F</td>
<td>5F</td>
<td>5F</td>
<td>5F</td>
<td>5F</td>
<td>5F</td>
<td>BA</td>
<td>BA</td>
<td>5F</td>
</tr>
<tr>
<td>`</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>51</td>
<td>DD</td>
<td>79</td>
<td>79</td>
<td>A0</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>C0</td>
<td>C0</td>
<td>C0</td>
<td>C0</td>
<td>43</td>
<td>9</td>
<td>43</td>
<td>44</td>
<td>C0</td>
<td>C0</td>
<td>51</td>
<td>C0</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 to 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 sort order corresponds directly to the arrangement of the code points within the code page. UTF-8 is the only SAS session encoding supported by SAS Viya.

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, and 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
- an encoding value
- a linguistic collation

*Table 3.4 on page 14* 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>
</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 côte.

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.

### 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), which 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, in the following statements, the SORT procedure collates linguistically, in accordance with the French_France locale:

```sas
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, the local conventions such as data formatting, and the 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” on page 599 or the SORTSEQ= option in the PROC SORT statement in SAS Viya Visual Data Management and Utility Procedures Guide.

A SAS data set that is sorted linguistically contains a sort indicator that displays the collating sequence LINGUISTIC in the 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, which is also displayed in the 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:

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

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

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>SASHHELP.CLASS</th>
<th>Observations 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables 5</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes 0</td>
</tr>
<tr>
<td>Created</td>
<td>01/17/2016 20:12:45</td>
<td>Observation Length 40</td>
</tr>
<tr>
<td>Last Modified</td>
<td>01/17/2016 20:12:45</td>
<td>Deleted Observations 0</td>
</tr>
<tr>
<td>Protection</td>
<td></td>
<td>Compressed NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td></td>
<td>Sorted NO</td>
</tr>
<tr>
<td>Label</td>
<td>Student Data</td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64</td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>us-ascii ASCII (ANSI)</td>
<td></td>
</tr>
</tbody>
</table>
Encoding Behavior in a SAS Session

Encoding Support for Data Sets by SAS Release

Data sets that are created in SAS automatically have UTF-8 set in the encoding attribute, which is in the descriptor portion of the file.

Output Processing

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

- If a new output file is created, the data is written to the current session encoding, UTF-8.
- 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. For more information, see “OUTREP= Data Set Option” on page 44.
- 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 specified, then the current session encoding is used.

Input Processing

For input (read) processing in SAS, the 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 encoding specified for the file is LATIN1, SAS Viya transcodes the file from LATIN1 to its session encoding of UTF-8.
- If a file does not have an encoding specified, SAS transcodes the data only if the file's data representation is different from the current session.

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. SAS Viya assumes that the encoding of the file is UTF-8. 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 20.
Chapter 4
Transcoding for NLS

Overview to 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. UTF-8 is the only SAS session encoding supported by SAS Viya.

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 and then moved to SAS Viya, which uses the UTF-8 encoding. When the file is processed on SAS Viya, the data is remapped from the LATIN1 encoding to the UTF-8 encoding.

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 more information, see Chapter 18, “Encoding Values in SAS Language Elements,” on page 643.
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 with an encoding other than UTF-8.

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>
<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>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 643.
Transcoding Considerations

Although transcoding usually occurs without problems, there are situations that can affect your data and produce unsatisfactory results. For example:

- 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 an 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.

- Incorrect encoding can be stamped on a SAS 7 or SAS 8 data set if it is copied or replaced in a SAS Viya session. 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.

Compatible and Incompatible Encodings

Overview to 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.

ASCII has been the foundation for most encodings, and it is used by most personal computers, minicomputers, and workstations. However, the IBM mainframe uses an EBCDIC encoding. Therefore, ASCII and EBCDIC machines and data are incompatible. Transcoding is necessary if some or all characters in one encoding are different from the characters in the other encoding.

UTF-8 is the only SAS session encoding supported by SAS Viya.

However, to avoid transcoding, you can create a data set and specify an encoding value that SAS does not transcode. For example, if you use the following values in either the
ENCODING= data set option or the INENCODING= or the OUTENCODING= option in the LIBNAME statement, transcoding is not performed:

- 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.

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

You might want to create a SAS data set that contains mixed encodings. An example is both Latin1 and Latin2. You do not want the data transcoded for either input or output processing. By default, data is transcoded to the current session encoding.

Data must be transcoded when the SAS file does not match the SAS Viya session encoding of UTF-8.

In some cases, transcoding is not required because the SAS file and the SAS session have compatible encodings.

**Line-feed Characters and Transferring Data between EBCDIC and ASCII**

Software that runs under ASCII operating environments requires that 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 an EBCDIC line-feed character to an 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 the line correctly, resulting in better formatting.

**Preventing Transcoding**

Some encoding values enable you to create a data set that SAS does not transcode. You might not want to transcode data for input or output processing. You might want to create a SAS library that contains data in mixed encodings instead. An example is both LATIN1 and LATIN2.

For example, you can avoid transcoding if you use the following values in either the ENCODING= data set option or the INENCODING= or OUTENCODING= option in the LIBNAME statement:

- 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 specifies that no transcoding is required between any ASCII-based encodings.
- EBCDICANY specifies that no transcoding is required between any EBCDIC-based encodings.

For more information, see “ENCODING= Data Set Option” on page 41 and “INENCODING= and OUTENCODING= Options” on page 615.

You can prevent transcoding for a specific column of data while the rest of the character data in the data set is transcoded by using the TRANSCODE= option. For more information, see “TRANSCODE= Column Modifier on PROC SQL” on page 618.

---

**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 (WLASTIN1), some of the characters that require 1 byte in WLASTIN1 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 1 byte, and a DBCS represents each character in 1 or 2 bytes. An MBCS represents characters in a varying length from 1 to 4 bytes. For example, when transcoding from WLASTIN2 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, and then the data is processed. Think of the CVP engine as an intermediate engine that prepares the data for transcoding. After the lengths are increased, the primary engine, such as the default base engine, actually processes the file.

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 use 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 `CVPENGINE=` option to specify the primary engine to process 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 multiples 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 CVP engine does not adjust formats or informats automatically. If you have attached formats to your data variables, you might have truncation issues. Adjust these formats manually or programmatically.

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

For more information and examples, see “`CVPBYTES=`, `CVPENGINE=`, and `CVPMULTIPLIER=` Options” on page 605.
Chapter 5
Time Zones

Overview of SAS Time Zone IDs

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) 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
Specifies 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 2, “Time Zone IDs and Time Zone Names,” on page 663.

Time zone name
Specifies three or four characters that indicate a time zone. For a list of time zone names, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 663.

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

Universal Coordinated Time (UTC)
Specifies 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 2, “Time Zone IDs and Time Zone Names,” on page 663.

When you specify a time zone, SAS adds a time-zone-specific timestamp to data sets and SAS catalogs when the data set or catalog is created or modified. The time-zone-
specific timestamp is also added to the SAS log when SAS starts and to the output at execution. 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.

### 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 2, “Time Zone IDs and Time Zone Names,” on page 663. For more information, see “TIMEZONE= System Option” on page 588.

#### 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:

```sas
data _null_;  
o1=tzoneoff();  
o2=tzoneoff('asia/tokyo');  
put o1 time.;  
put o2 time.;  
run;
```

Here is the output in the SAS log:

```
-5:00:00
9:00:00
```

To find the difference between two time zones, you can use the `ABS(_)` function:

```sas
diff=abs(tzoneoff('america/new_york') - tzoneoff('asia/tokyo'));
```

For more information, see “TZONEOFF Function” on page 414.

#### 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:

    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 zone for Los Angeles daylight   |
| Time zone name for this SAS          |

For more information, see “TZONENAME Function” on page 412.

**Convert Datetime Values between SAS and UTC**

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

• The TZONES2U( ) 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:

    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 2013-03-15T09:15:00+00:00 ;
    put ' ';
    put ' Change a SAS datetime to a UTC value ;
    put ' ';
    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 ' ';
    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';
Here is the output:

<table>
<thead>
<tr>
<th>The time zone is EST</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York-London difference: 5:00:00</td>
</tr>
<tr>
<td>New York-Tokyo difference: 14:00:00</td>
</tr>
<tr>
<td>The SAS datetime is 2013-03-15T09:15:00+00:00</td>
</tr>
</tbody>
</table>

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 415.

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’s 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’s 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’s local time zone.

This program formats time and datetime values based on the time zone ID Australia/Sydney:
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: 2013-03-17 19:30:22 +0500
SAS datetime NLDATMWZ: Sunday, March 17, 2013 07:30:22 PM +0500
```

For more information, see “NLDATMTZ Format” on page 138, “NLDATMZ Format” on page 151, and “NLDATMWZ Format” on page 141.
Time Zone Example

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

/* 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 nldatmwz.;
/* 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:
/* 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 nldatmwz.;
   /* Set the flight time */
   ftime='13:00't;
   put 'Flight time=' ftime time.;
   utc=depart+ftime;
   put 'Arrive PST=' utc nldatmwz.;
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
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
/* Set the time zone for Tokyo */
options timezone='Asia/Tokyo';
data arrive;
   set depart;
   put 'Arrive in Tokyo ' utc nldatmwz.;
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
Part 2

Autocall Macros for NLS

Chapter 6
Dictionary of Autocall Macros for NLS .......................... 35
Chapter 6
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 and %QKLOWCAS Autocall Macros (p. 35)</td>
<td>Change uppercase characters to lowercase.</td>
</tr>
<tr>
<td></td>
<td>%KTRIM and %QKTRIM Autocall Macros (p. 36)</td>
<td>Trim trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%KVERIFY Autocall Macro (p. 36)</td>
<td>Returns the position of the first character unique to an expression.</td>
</tr>
</tbody>
</table>

Dictionary

%KLOWCASE and %QKLOWCAS Autocall Macros
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 and %QKTRIM Autocall Macros

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.
Part 3

Data Set Options for NLS

Chapter 7

Dictionary of Data Set Options for NLS ........................................ 41
Chapter 7
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. 41)</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. 44)</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
Restriction: UTF-8 is the only SAS session encoding supported by SAS Viya.
Syntax

ENCODING= ANY | ASCIIANY | 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.

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. UTF-8 is the only SAS session encoding supported by SAS Viya.

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. UTF-8 is the only SAS session encoding supported by SAS Viya.

- 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);
  set work.latin1;
  set work.latin2;
run;
```

**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 data set is being created for use in a SAS environment other than SAS Viya.

For example, if the current session encoding is UTF-8, 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.
encoding instead of the session encoding. The encoding is also specified in the descriptor portion of the file.

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

See Also

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

Options in Statements and Commands:

• “ENCODING= Option” on page 612
• “INENCODING= and OUTENCODING= Options” on page 615

System Options:

• “ENCODING System Option” on page 577
• “LOCALE System Option” on page 578

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 Viya Data Set Options: Reference.

Syntax

OUTREP= format

See Also

“OUTREP= Data Set Option” in SAS Viya Data Set Options: Reference
Part 4

Formats for NLS

Chapter 8
Overview to NLS Formats ......................................................... 47

Chapter 9
Dictionary of Formats for NLS .................................................... 63
Chapter 8  
Overview to NLS Formats

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>
<thead>
<tr>
<th>Language</th>
<th>English Format</th>
<th>International Format</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEG.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>---------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEN.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEN.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEN.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>French (FRA)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>German (DEU)</td>
<td>DATE.</td>
<td>105</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Hungarian (HUN)

| DATE.                | NLDATE.        | 10                   | 200 | 20  |         |
| DATETIME.            | NLDATM.        | 10                   | 200 | 30  |         |
| DOWNNAME.            | NLDATEWN.      | 4                    | 200 | 9   |         |
| MONNAME.             | NLDATEMN.      | 4                    | 200 | 9   |         |
| MONYY.               | NLDATEYM.      | 6                    | 200 | 16  |         |
| WEEKDATX.            | NLDATEW.       | 10                   | 200 | 29  |         |
| WEEKDAY.             | NLDATEWN.      | 4                    | 200 | 9   |         |
| WORDDATX.            | NLDATE.        | 10                   | 200 | 20  |         |

Italian (ITA)

| DATE.                | NLDATE.        | 10                   | 200 | 20  |         |
| DATETIME.            | NLDATM.        | 10                   | 200 | 30  |         |
| DOWNNAME.            | NLDATEWN.      | 4                    | 200 | 9   |         |
| MONNAME.             | NLDATEMN.      | 4                    | 200 | 9   |         |
| MONYY.               | NLDATEYM.      | 6                    | 200 | 16  |         |
| WEEKDATX.            | NLDATEW.       | 10                   | 200 | 29  |         |
| WEEKDAY.             | NLDATEWN.      | 4                    | 200 | 9   |         |
| WORDDATX.            | NLDATE.        | 10                   | 200 | 20  |         |

Macedonian (MAC)

<p>| DATE.                | NLDATE.        | 10                   | 200 | 20  |         |
| DATETIME.            | NLDATM.        | 10                   | 200 | 30  |         |
| DOWNNAME.            | NLDATEWN.      | 4                    | 200 | 9   |         |
| MONNAME.             | NLDATEMN.      | 4                    | 200 | 9   |         |
| MONYY.               | NLDATEYM.      | 6                    | 200 | 16  |         |
| WEEKDATX.            | NLDATEW.       | 10                   | 200 | 29  |         |</p>
<table>
<thead>
<tr>
<th>Language</th>
<th>English Format</th>
<th>International Format</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WEEKDDATX.</td>
<td>EURDFWDX.</td>
<td>1</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATEWX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></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 it 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

• lack of support for all currency symbols is a limitation
• reversal of the dot and comma for currency formatting is not used by all European countries
• 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_; x=12345; put x euro10.2; run;

Output:

E12,345.00
Euro format is not locale–sensitive. It uses a comma(,) as the thousands separator and a dot (.) as the decimal separator.

**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 more information about the FORMAT procedure, see *SAS Viya Visual Data Management and Utility Procedures Guide*.

**Example Code 8.1  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;

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

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 they require 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:

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

Output:

$12,345.00

Selected national currency representations follow:

<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>
<tr>
<td>Spanish_Venezuela</td>
<td>Venezuelan bolivars</td>
<td>VEF12.345,00</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. French_France uses no thousands separator but uses a comma as a decimal separator, while 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 NLMNYIw.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). You must also use the LOCALE= option to specify the locale when using
the NLMNYIw.d formats and informats. The letter “I,” which signifies “International,” is appended to the format and informat names. Here is an example:

```plaintext
options locale=english_UnitedStates;
data _null_;  
x=12345; put x nlmnyi15.2;  
run;  
Output:  
USD12,345.00
```

Example:

Selected international currency representations follow:

<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>CHF 12'345.00</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 and a dot as the decimal separator. When used, the ISO currency code placement varies by locale. The EMU countries put the currency code after the currency, while 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 NLMNLISOw.d and NLMNIISOw.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.
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_; x=12345; 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.

Example: `data _null_; x=12345; put x nlmnleur15.2; run;`

Output: **€12,345.00**

<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 used are a comma 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 NLMNLxxxxw.d format formats the value with the three-letter ISO currency code.

Unique international monetary representation is specified by the unique ISO currency code. International formats are specified using the NLMNI/ISOw.d formats and informats. International formats 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 NLMNI/ISOw.d formats and informats, do not use the LOCALE= option to specify the locale.
Example: `data _null_; x=12345; put x nlmniusd15.2; run;` Output: USD12,345.00

Selected international currency representations follow:

<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 8.2 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;
```

58 Chapter 8 • Overview to NLS Formats
These exchange rates, which were effective June 17, 2004, are specified as data in the SAS program.

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.

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 8.1 National and International Format Output
### Exceptions for Date and Time Default Widths

This table lists the recommended widths for UTF-8.

<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>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></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></td>
<td>JA_JP</td>
<td>nldatmw</td>
<td>30</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldateyq</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>KO_KR</td>
<td>nldatew</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmw</td>
<td>30</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>RU_RU</td>
<td>nldatew</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmw</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>ZH_CN</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmw</td>
<td>30</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>ZT_HK</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmw</td>
<td>30</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>ZT_TW</td>
<td>nldatew</td>
<td>20</td>
<td>25</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>nldatmw</td>
<td></td>
<td></td>
<td>30</td>
<td>49</td>
</tr>
<tr>
<td>nldatmwz</td>
<td></td>
<td></td>
<td>40</td>
<td>55</td>
</tr>
<tr>
<td>nltimap</td>
<td></td>
<td></td>
<td>10</td>
<td>21</td>
</tr>
</tbody>
</table>
Chapter 8 • Overview to NLS Formats
Chapter 9
Dictionary of Formats for NLS

Categories of NLS Formats ......................................................... 66
Dictionary ................................................................. 82
B8601DX Format .............................................................. 82
B8601LX Format .............................................................. 84
B8601TX Format .............................................................. 85
BESTDOTX Format ............................................................ 87
$CPTDW Format .............................................................. 88
$CPTWD Format .............................................................. 89
$EBCDIC Format .............................................................. 90
E8601DX Format .............................................................. 91
E8601LX Format .............................................................. 93
E8601TX Format .............................................................. 94
EURO Format ................................................................. 96
EUROX Format ............................................................... 99
MINGUO Format ............................................................ 101
NENGO Format ............................................................. 102
NLBEST Format ............................................................ 104
NLDATE Format ............................................................ 106
NLDATEL Format ........................................................... 107
NLDATEM Format .......................................................... 108
NLDATEMD Format ......................................................... 109
NLDATEMDL Format ......................................................... 110
NLDATEMDM Format ....................................................... 111
NLDATEMS Format ........................................................ 111
NLDATEMN Format ........................................................ 112
NLDATES Format ........................................................... 113
NLDATEW Format .......................................................... 114
NLDATEWN Format ......................................................... 116
NLDATEYM Format ........................................................ 117
NLDATEYML Format ......................................................... 118
NLDATEYMM Format ....................................................... 119
NLDATEYMS Format ....................................................... 120
NLDATEYQ Format ........................................................ 121
NLDATEYQL Format ....................................................... 122
NLDATEYQM Format ....................................................... 123
NLDATEYQS Format ....................................................... 124
NLDATEYR Format ........................................................ 125
NLDATEYW Format ........................................................ 126
NLDATM Format ............................................................ 127
NLDATMAP Format ......................................................... 128
NLDATMDT Format ......................................................... 129
NLNITHB Format .......................................................... 184
NLNITRY Format .......................................................... 185
NLNITWD Format .......................................................... 186
NLNIUSD Format .......................................................... 187
NLNIZAR Format .......................................................... 188
NLNLNLTWD Format ...................................................... 189
NLNLAUD Format .......................................................... 190
NLNLBGN Format .......................................................... 191
NLNLBRL Format .......................................................... 192
NLNLCAD Format .......................................................... 193
NLNLCHF Format .......................................................... 194
NLNLCNY Format .......................................................... 195
NLNLČZK Format .......................................................... 196
NLNLĐKK Format .......................................................... 197
NLNLEEK Format .......................................................... 198
NLNLLEG Format .......................................................... 199
NLNLEUR Format .......................................................... 200
NLNLGBP Format .......................................................... 201
NLNLHKD Format .......................................................... 202
NLNLHRK Format .......................................................... 203
NLNLHUF Format .......................................................... 204
NLNLIDR Format .......................................................... 205
NLNLILS Format .......................................................... 206
NLNLINR Format .......................................................... 207
NLNLJPY Format .......................................................... 208
NLNLKRW Format .......................................................... 209
NLNLKKL Format .......................................................... 210
NLNLKKL Format .......................................................... 211
NLNLKZL Format .......................................................... 212
NLNLKZL Format .......................................................... 213
NLNLKZL Format .......................................................... 214
NLNLKZL Format .......................................................... 215
NLNLFNOK Format .......................................................... 216
NLNLNZD Format .......................................................... 217
NLNLNPLN Format .......................................................... 218
NLNLPLN Format .......................................................... 219
NLNLPLN Format .......................................................... 220
NLNLPLN Format .......................................................... 221
NLNLTRY Format .......................................................... 222
NLNLTRY Format .......................................................... 223
NLNLUSD Format .......................................................... 224
NLNLNZAR Format .......................................................... 225
NLNMN Format .............................................................. 226
NLNMN Format .............................................................. 227
NLNMN Format .............................................................. 228
NLNMN Format .............................................................. 230
NLPCT Format .............................................................. 231
NLPCTI Format .............................................................. 232
NLPCTN Format .............................................................. 233
NLPCTP Format .............................................................. 234
NLPVALUE Format .......................................................... 235
NLSSTRMON Format ...................................................... 236
NLSSTRQTR Format ...................................................... 237
NLSSTRWK Format ...................................................... 238
NLSSTRWK Format ...................................................... 239
NLSUTIL Format ............................................................. 240
NLSUTIL Format ............................................................. 241
NLSUTIL Format ............................................................. 242
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.

The following categories relate 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>Category</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td>EURO Format (p. 96)</td>
<td>Writes numeric values with a leading euro symbol (E), a comma that separates every three digits, and a period that separates the decimal fraction.</td>
</tr>
<tr>
<td></td>
<td>EUROX Format (p. 99)</td>
<td>Writes numeric values with a leading euro symbol (E), a period that separates every three digits, and a comma that separates the decimal fraction.</td>
</tr>
<tr>
<td></td>
<td>NENGO Format (p. 102)</td>
<td>Writes date values as Japanese dates in the form e.yymmdd.</td>
</tr>
<tr>
<td></td>
<td>NLBEST Format (p. 104)</td>
<td>Writes the best numerical notation based on the locale.</td>
</tr>
<tr>
<td></td>
<td>NLDATE Format (p. 106)</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date.</td>
</tr>
<tr>
<td></td>
<td>NLDATEL Format (p. 107)</td>
<td>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, date, month, year.</td>
</tr>
<tr>
<td></td>
<td>NLDATEM Format (p. 108)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMD Format (p. 109)</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMDL Format (p. 110)</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.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMDM Format (p. 111)</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.</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>NLDATEMDS Format (p. 111)</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.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMNN Format (p. 112)</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the value as the name of the month.</td>
</tr>
<tr>
<td></td>
<td>NLDATES Format (p. 113)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date string.</td>
</tr>
<tr>
<td></td>
<td>NLDATENW Format (p. 114)</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></td>
<td>NLDATENWN Format (p. 116)</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></td>
<td>NLDATENYM Format (p. 117)</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></td>
<td>NLDATENYML Format (p. 118)</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></td>
<td>NLDATENYMM Format (p. 119)</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></td>
<td>NLDATENYMS Format (p. 120)</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></td>
<td>NLDATENYQ Format (p. 121)</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></td>
<td>NLDATENYQL Format (p. 122)</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></td>
<td>NLDATENYQM Format (p. 123)</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></td>
<td>NLDATENYQS Format (p. 124)</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></td>
<td>NLDATENYR Format (p. 125)</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year.</td>
</tr>
<tr>
<td></td>
<td>NLDATENYW Format (p. 126)</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></td>
<td>NLDATNM Format (p. 127)</td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLDATMAP Format</td>
<td>(p. 128)</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>NLDATMDT Format</td>
<td>(p. 129)</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>NLDATML Format</td>
<td>(p. 130)</td>
<td>Converts a SAS datetime 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.</td>
</tr>
<tr>
<td>NLDATMM Format</td>
<td>(p. 131)</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 time with abbreviations for the month and time.</td>
</tr>
<tr>
<td>NLDATMMD Format</td>
<td>(p. 131)</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>NLDATMMDL Format</td>
<td>(p. 132)</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>NLDATMMDM Format</td>
<td>(p. 133)</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>NLDATMMDS Format</td>
<td>(p. 134)</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>NLDATMMN Format</td>
<td>(p. 135)</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>NLDATMS Format</td>
<td>(p. 136)</td>
<td>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 .</td>
</tr>
<tr>
<td>NLDATMTM Format</td>
<td>(p. 137)</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>NLDATMTZ Format</td>
<td>(p. 138)</td>
<td>Converts the time portion of the SAS datetime of the locale to the time of day and time zone.</td>
</tr>
<tr>
<td>NLDATMW Format</td>
<td>(p. 139)</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>NLDATMWN Format</td>
<td>(p. 140)</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>NLDATMWZ Format</td>
<td>(p. 141)</td>
<td>Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone 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>NLDATMYM Format (p. 142)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the month.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYML Format (p. 143)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYMM Format (p. 144)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYMS Format (p. 144)</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 delimiters.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQ Format (p. 145)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQL Format (p. 146)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYQM Format (p. 147)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYQS Format (p. 148)</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 delimiters.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYR Format (p. 149)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYW Format (p. 150)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.</td>
<td></td>
</tr>
<tr>
<td>NLDATMZ Format (p. 151)</td>
<td>Converts SAS datetime values to the locale-sensitive datetime string as time zone and datetime.</td>
<td></td>
</tr>
<tr>
<td>NLMNY Format (p. 226)</td>
<td>Writes the monetary format of the local expression in the specified locale using local currency.</td>
<td></td>
</tr>
<tr>
<td>NLMNYI Format (p. 227)</td>
<td>Writes the monetary format of the international expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLNUM Format (p. 228)</td>
<td>Writes the numeric format of the local expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLNUMI Format (p. 230)</td>
<td>Writes the numeric format of the international expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLPCT Format (p. 231)</td>
<td>Writes percentage data of the local expression in the specified locale.</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>NLPCTI Format (p. 232)</td>
<td></td>
<td>Writes percentage data of the international expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCTN Format (p. 234)</td>
<td></td>
<td>Produces percentages, using a minus sign for negative values.</td>
</tr>
<tr>
<td>NLPCTP Format (p. 235)</td>
<td></td>
<td>Writes locale-specific numeric values as percentages.</td>
</tr>
<tr>
<td>NLPVALUE Format (p. 236)</td>
<td></td>
<td>Writes p-values of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLSTRMON Format (p. 237)</td>
<td></td>
<td>Writes the month name in the specified locale.</td>
</tr>
<tr>
<td>NLSTRQTR Format (p. 238)</td>
<td></td>
<td>Writes a numeric value as the quarter-of-the-year in the specified locale.</td>
</tr>
<tr>
<td>NLSTRWK Format (p. 239)</td>
<td></td>
<td>Writes a numeric value as the day-of-the-week in the specified locale.</td>
</tr>
<tr>
<td>NLTIMAP Format (p. 241)</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>NLTIME Format (p. 242)</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>YEN Format (p. 271)</td>
<td></td>
<td>Writes numeric values with yen signs, commas, and decimal points.</td>
</tr>
<tr>
<td>YYWEEKU Format (p. 272)</td>
<td></td>
<td>Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.</td>
</tr>
<tr>
<td>YYWEEKV Format (p. 273)</td>
<td></td>
<td>Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.</td>
</tr>
<tr>
<td>YYWEEKW Format (p. 275)</td>
<td></td>
<td>Writes a week number in decimal format by using the W algorithm, excluding the day-of-week information.</td>
</tr>
<tr>
<td>Character</td>
<td>$EBCDIC Format (p. 90)</td>
<td>Converts native format character data to EBCDIC representation.</td>
</tr>
<tr>
<td></td>
<td>SUCS2B Format (p. 243)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>SUCS2BE Format (p. 244)</td>
<td>Processes a character string that is in big-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>SUCS2L Format (p. 245)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>SUCS2LE Format (p. 247)</td>
<td>Processes a character string that is in little-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$UCS2X Format (p. 248)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>$UCS2XE Format (p. 249)</td>
<td>Processes a character string that is in native-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4B Format (p. 250)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>$UCS4BE Format (p. 251)</td>
<td>Processes a character string that is in big-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4L Format (p. 252)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>$UCS4LE Format (p. 254)</td>
<td>Processes a character string that is in little-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4X Format (p. 255)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>$UCS4XE Format (p. 256)</td>
<td>Processes a character string that is in native-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UESC Format (p. 257)</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>
<td></td>
</tr>
<tr>
<td>$UESCE Format (p. 259)</td>
<td>Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UNCR Format (p. 260)</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>
<td></td>
</tr>
<tr>
<td>$UNCRE Format (p. 261)</td>
<td>Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UPAREN Format (p. 262)</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>
<td></td>
</tr>
<tr>
<td>$UPARENE Format (p. 263)</td>
<td>Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$UTF8X Format (p. 264)</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>
<td></td>
</tr>
<tr>
<td>$UTF8XE Format (p. 265)</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>
<td></td>
</tr>
<tr>
<td>Date and Time</td>
<td>B8601DX Format (p. 82)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone basic notation yyyymmdTthhmmss+hhmm.</td>
</tr>
<tr>
<td></td>
<td>B8601LX Format (p. 84)</td>
<td>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 yyyymmdTthhmmss+</td>
</tr>
<tr>
<td></td>
<td>B8601TX Format (p. 85)</td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 basic time notation hhmmss+</td>
</tr>
<tr>
<td></td>
<td>E8601DX Format (p. 91)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user 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.</td>
</tr>
<tr>
<td></td>
<td>E8601LX Format (p. 93)</td>
<td>Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 extended notation yyyy-mm-ddThh:mm:ss+</td>
</tr>
<tr>
<td></td>
<td>E8601TX Format (p. 94)</td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 extended time notation hh:mm:ss+</td>
</tr>
<tr>
<td></td>
<td>MINGUO Format (p. 101)</td>
<td>Writes date values as Taiwanese dates in the form yyyymmd.</td>
</tr>
<tr>
<td></td>
<td>NENGO Format (p. 102)</td>
<td>Writes date values as Japanese dates in the form e.yymmdd.</td>
</tr>
<tr>
<td></td>
<td>NLDATE Format (p. 106)</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date.</td>
</tr>
<tr>
<td></td>
<td>NLDATEL Format (p. 107)</td>
<td>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, date, month, year.</td>
</tr>
<tr>
<td></td>
<td>NLDATEM Format (p. 108)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMD Format (p. 109)</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMDL Format (p. 110)</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.</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>NLDATEMDM Format (p. 111)</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.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMDS Format (p. 111)</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.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMN Format (p. 112)</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the value as the name of the month.</td>
</tr>
<tr>
<td></td>
<td>NLDATES Format (p. 113)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date string.</td>
</tr>
<tr>
<td></td>
<td>NLDATEW Format (p. 114)</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></td>
<td>NLDATEWN Format (p. 116)</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></td>
<td>NLDATEYM Format (p. 117)</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></td>
<td>NLDATEYML Format (p. 118)</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></td>
<td>NLDATEYMM Format (p. 119)</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></td>
<td>NLDATEYMS Format (p. 120)</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></td>
<td>NLDATEYQ Format (p. 121)</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></td>
<td>NLDATEYQL Format (p. 122)</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></td>
<td>NLDATEYQM Format (p. 123)</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></td>
<td>NLDATEYQS Format (p. 124)</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></td>
<td>NLDATEYR Format (p. 125)</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLDATEYW Format (p. 126)</td>
<td></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>NLDATM Format (p. 127)</td>
<td></td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.</td>
</tr>
<tr>
<td>NLDATMAP Format (p. 128)</td>
<td></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>NLDATMDT Format (p. 129)</td>
<td></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>NLDATML Format (p. 130)</td>
<td></td>
<td>Converts a SAS datetime 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.</td>
</tr>
<tr>
<td>NLDATMM Format (p. 131)</td>
<td></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 time with abbreviations for the month and time.</td>
</tr>
<tr>
<td>NLDATMMD Format (p. 131)</td>
<td></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>NLDATMMDL Format (p. 132)</td>
<td></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>NLDATMMDM Format (p. 133)</td>
<td></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>NLDATMMDS Format (p. 134)</td>
<td></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>NLDATMMN Format (p. 135)</td>
<td></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>NLDATMS Format (p. 136)</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>NLDATMTM Format (p. 137)</td>
<td></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>NLDATMTZ Format (p. 138)</td>
<td></td>
<td>Converts the time portion of the SAS datetime of the locale to the time of day and time zone.</td>
</tr>
<tr>
<td>NLDATMW Format (p. 139)</td>
<td></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>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLDATMWN Format (p. 140)</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>
<td></td>
</tr>
<tr>
<td>NLDATMWZ Format (p. 141)</td>
<td>Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYM Format (p. 142)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the month.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYML Format (p. 143)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYMM Format (p. 144)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYMS Format (p. 144)</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 delimiters.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQ Format (p. 145)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQL Format (p. 146)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYQM Format (p. 147)</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>
<td></td>
</tr>
<tr>
<td>NLDATMYQS Format (p. 148)</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 delimiters.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYR Format (p. 149)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYW Format (p. 150)</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.</td>
<td></td>
</tr>
<tr>
<td>NLDATMZ Format (p. 151)</td>
<td>Converts SAS datetime values to the locale-sensitive datetime string as time zone and datetime.</td>
<td></td>
</tr>
<tr>
<td>NLTIMAP Format (p. 241)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>NLTIME Format (p. 242)</td>
<td>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.</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>WEEKU Format (p. 266)</td>
<td>Writes a week number in decimal format by using the U algorithm.</td>
</tr>
<tr>
<td></td>
<td>WEEKV Format (p. 267)</td>
<td>Writes a week number in decimal format by using the V algorithm.</td>
</tr>
<tr>
<td></td>
<td>WEEKW Format (p. 269)</td>
<td>Writes a week number in decimal format by using the W algorithm.</td>
</tr>
<tr>
<td></td>
<td>YYWEEKU Format (p. 272)</td>
<td>Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.</td>
</tr>
<tr>
<td></td>
<td>YYWEEKV Format (p. 273)</td>
<td>Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.</td>
</tr>
<tr>
<td></td>
<td>YYWEEKW Format (p. 275)</td>
<td>Writes a week number in decimal format by using the W algorithm, excluding the day-of-week information.</td>
</tr>
<tr>
<td>Hebrew Text Handling</td>
<td>$CPTDW Format (p. 88)</td>
<td>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).</td>
</tr>
<tr>
<td></td>
<td>$CPTWD Format (p. 89)</td>
<td>Processes a character string that is encoded in Windows (cp1255), and then writes the character string in Hebrew DOS (cp862) encoding.</td>
</tr>
<tr>
<td>ISO 8601</td>
<td>B8601DX Format (p. 82)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone basic notation yyyymmddThhmmss+hhmm.</td>
</tr>
<tr>
<td></td>
<td>B8601LX Format (p. 84)</td>
<td>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 yyyymmddThhmmss+-hhmm.</td>
</tr>
<tr>
<td></td>
<td>B8601TX Format (p. 85)</td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 basic time notation hh:mm:ss+-hh:mm.</td>
</tr>
<tr>
<td></td>
<td>E8601DX Format (p. 91)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user 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.</td>
</tr>
<tr>
<td></td>
<td>E8601LX Format (p. 93)</td>
<td>Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 extended notation yyyy-mm-ddThh:mm:ss+-hh:mm.</td>
</tr>
<tr>
<td></td>
<td>E8601TX Format (p. 94)</td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 extended time notation hh:mm:ss+-hh:mm.</td>
</tr>
<tr>
<td>Numeric</td>
<td>BESTDOTX Format (p. 87)</td>
<td>Specifies that SAS choose the best notation and use a dot as a decimal separator.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>EURO Format (p. 96)</td>
<td>Writes numeric values with a leading euro symbol (€), a comma that separates every three digits, and a period that separates the decimal fraction.</td>
<td></td>
</tr>
<tr>
<td>EUROX Format (p. 99)</td>
<td>Writes numeric values with a leading euro symbol (€), a period that separates every three digits, and a comma that separates the decimal fraction.</td>
<td></td>
</tr>
<tr>
<td>NLBEST Format (p. 104)</td>
<td>Writes the best numerical notation based on the locale.</td>
<td></td>
</tr>
<tr>
<td>NLMNIAED Format (p. 152)</td>
<td>Writes the monetary format of the international expression for the United Arab Emirates.</td>
<td></td>
</tr>
<tr>
<td>NLMNIAUD Format (p. 153)</td>
<td>Writes the monetary format of the international expression for Australia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIBGN Format (p. 154)</td>
<td>Writes the monetary format of the international expression for Bulgaria.</td>
<td></td>
</tr>
<tr>
<td>NLMNIBRL Format (p. 155)</td>
<td>Writes the monetary format of the international expression for Brazil.</td>
<td></td>
</tr>
<tr>
<td>NLMNICAD Format (p. 156)</td>
<td>Writes the monetary format of the international expression for Canada.</td>
<td></td>
</tr>
<tr>
<td>NLMNICHF Format (p. 157)</td>
<td>Writes the monetary format of the international expression for Liechtenstein and Switzerland.</td>
<td></td>
</tr>
<tr>
<td>NLMNICNY Format (p. 158)</td>
<td>Writes the monetary format of the international expression for China.</td>
<td></td>
</tr>
<tr>
<td>NLMNICZK Format (p. 159)</td>
<td>Writes the monetary format of the international expression for the Czech Republic.</td>
<td></td>
</tr>
<tr>
<td>NLMNIDKK Format (p. 160)</td>
<td>Writes the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
<td></td>
</tr>
<tr>
<td>NLMNIEEK Format (p. 161)</td>
<td>Writes the monetary format of the international expression for Estonia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIEGP Format (p. 162)</td>
<td>Writes the monetary format of the international expression for Egypt.</td>
<td></td>
</tr>
<tr>
<td>NLMNIEUR Format (p. 163)</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>
<td></td>
</tr>
<tr>
<td>NLMNIGBP Format (p. 164)</td>
<td>Writes the monetary format of the international expression for the United Kingdom.</td>
<td></td>
</tr>
<tr>
<td>NLMNIIHKD Format (p. 165)</td>
<td>Writes the monetary format of the international expression for Hong Kong.</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>NLMNIHRK Format (p. 166)</td>
<td>Writes the monetary format of the international expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIHUF Format (p. 167)</td>
<td>Writes the monetary format of the international expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>NLMNIIDR Format (p. 168)</td>
<td>Writes the monetary format of the international expression for Indonesia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIILS Format (p. 169)</td>
<td>Writes the monetary format of the international expression for Israel.</td>
<td></td>
</tr>
<tr>
<td>NLMNIINR Format (p. 170)</td>
<td>Writes the monetary format of the international expression for India.</td>
<td></td>
</tr>
<tr>
<td>NLMNIJJPY Format (p. 171)</td>
<td>Writes the monetary format of the international expression for Japan.</td>
<td></td>
</tr>
<tr>
<td>NLMNIKRW Format (p. 172)</td>
<td>Writes the monetary format of the international expression for South Korea.</td>
<td></td>
</tr>
<tr>
<td>NLMNILTL Format (p. 173)</td>
<td>Writes the monetary format of the international expression for Lithuania.</td>
<td></td>
</tr>
<tr>
<td>NLMNILVL Format (p. 174)</td>
<td>Writes the monetary format of the international expression for Latvia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMOP Format (p. 175)</td>
<td>Writes the monetary format of the international expression for Macau.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMXN Format (p. 176)</td>
<td>Writes the monetary format of the international expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMYR Format (p. 177)</td>
<td>Writes the monetary format of the international expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>NLMNINOK Format (p. 178)</td>
<td>Writes the monetary format of the international expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>NLMNINZD Format (p. 179)</td>
<td>Writes the monetary format of the international expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>NLMNIPLN Format (p. 180)</td>
<td>Writes the monetary format of the international expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>NLMNIRUB Format (p. 181)</td>
<td>Writes the monetary format of the international expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>NLMNISEK Format (p. 182)</td>
<td>Writes the monetary format of the international expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>NLMNISGD Format (p. 183)</td>
<td>Writes 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>NLMNITHB Format (p. 184)</td>
<td>Writes the monetary format of the international expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>NLMNITRY Format (p. 185)</td>
<td>Writes the monetary format of the international expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>NLMNITWD Format (p. 186)</td>
<td>Writes the monetary format of the international expression for Taiwan.</td>
<td></td>
</tr>
<tr>
<td>NLMNIUSD Format (p. 187)</td>
<td>Writes the monetary format of the international expression for Puerto Rico and the United States.</td>
<td></td>
</tr>
<tr>
<td>NLMNIZAR Format (p. 188)</td>
<td>Writes the monetary format of the international expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>NLMNLAED Format (p. 189)</td>
<td>Writes the monetary format of the local expression for the United Arab Emirates.</td>
<td></td>
</tr>
<tr>
<td>NLMNLAUD Format (p. 190)</td>
<td>Writes the monetary format of the local expression for Australia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLBGN Format (p. 191)</td>
<td>Writes the monetary format of the local expression for Bulgaria.</td>
<td></td>
</tr>
<tr>
<td>NLMNLBRL Format (p. 192)</td>
<td>Writes the monetary format of the local expression for Brazil.</td>
<td></td>
</tr>
<tr>
<td>NLMNLCAD Format (p. 193)</td>
<td>Writes the monetary format of the local expression for Canada.</td>
<td></td>
</tr>
<tr>
<td>NLMNLCHF Format (p. 194)</td>
<td>Writes the monetary format of the local expression for Liechtenstein and Switzerland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLCNY Format (p. 195)</td>
<td>Writes the monetary format of the local expression for China.</td>
<td></td>
</tr>
<tr>
<td>NLMNLČZK Format (p. 196)</td>
<td>Writes the monetary format of the local expression for the Czech Republic.</td>
<td></td>
</tr>
<tr>
<td>NLMNLDKK Format (p. 197)</td>
<td>Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLEEK Format (p. 198)</td>
<td>Writes the monetary format of the local expression for Estonia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLLEGP Format (p. 199)</td>
<td>Writes the monetary format of the local expression for Egypt.</td>
<td></td>
</tr>
<tr>
<td>NLMNLEUR Format (p. 200)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>NLMNLGBP Format (p. 201)</td>
<td>Writes the monetary format of the local expression for the United Kingdom.</td>
<td></td>
</tr>
<tr>
<td>NLMNLHKD Format (p. 202)</td>
<td>Writes the monetary format of the local expression for Hong Kong.</td>
<td></td>
</tr>
<tr>
<td>NLMNLHRK Format (p. 203)</td>
<td>Writes the monetary format of the local expression for Croatia.</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>NLMNLHUF Format (p. 204)</td>
<td>Writes the monetary format of the local expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>NLMNLIDR Format (p. 205)</td>
<td>Writes the monetary format of the local expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLILS Format (p. 206)</td>
<td>Writes the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMLINR Format (p. 207)</td>
<td>Writes the monetary format of the local expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNLJPY Format (p. 208)</td>
<td>Writes the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLKRW Format (p. 209)</td>
<td>Writes the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLTL Format (p. 210)</td>
<td>Writes the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLVL Format (p. 211)</td>
<td>Writes the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMOP Format (p. 212)</td>
<td>Writes the monetary format of the local expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMXN Format (p. 213)</td>
<td>Writes the monetary format of the local expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMYR Format (p. 214)</td>
<td>Writes the monetary format of the local expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNOK Format (p. 215)</td>
<td>Writes the monetary format of the local expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNZD Format (p. 216)</td>
<td>Writes the monetary format of the local expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNPLPLN Format (p. 217)</td>
<td>Writes the monetary format of the local expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLRUB Format (p. 218)</td>
<td>Writes the monetary format of the local expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLSEK Format (p. 219)</td>
<td>Writes the monetary format of the local expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>NLMNLSGD Format (p. 220)</td>
<td>Writes the monetary format of the local expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTHB Format (p. 221)</td>
<td>Writes the monetary format of the local expression for Thailand.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTRY Format (p. 222)</td>
<td>Writes the monetary format of the local expression for Turkey.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTWD Format (p. 223)</td>
<td>Writes the monetary format of the local expression for Taiwan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLUSD Format (p. 224)</td>
<td>Writes the monetary format of the local expression for Puerto Rico and the United States.</td>
</tr>
<tr>
<td></td>
<td>NLMNLZAR Format (p. 225)</td>
<td>Writes the monetary format of the local expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>NLMNY Format (p. 226)</td>
<td>Writes the monetary format of the local expression in the specified locale using local currency.</td>
</tr>
</tbody>
</table>
### Category

<table>
<thead>
<tr>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLMNYI Format (p. 227)</td>
<td>Writes the monetary format of the international expression in the specified locale.</td>
</tr>
<tr>
<td>NLNUM Format (p. 228)</td>
<td>Writes the numeric format of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLNUMI Format (p. 230)</td>
<td>Writes the numeric format of the international expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCT Format (p. 231)</td>
<td>Writes percentage data of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCTI Format (p. 232)</td>
<td>Writes percentage data of the international expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCTN Format (p. 234)</td>
<td>Produces percentages, using a minus sign for negative values.</td>
</tr>
<tr>
<td>NLPCTP Format (p. 235)</td>
<td>Writes locale-specific numeric values as percentages.</td>
</tr>
<tr>
<td>NLPVALUE Format (p. 236)</td>
<td>Writes p-values of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLSTRMON Format (p. 237)</td>
<td>Writes the month name in the specified locale.</td>
</tr>
<tr>
<td>NLSTRQTR Format (p. 238)</td>
<td>Writes a numeric value as the quarter-of-the-year in the specified locale.</td>
</tr>
<tr>
<td>NLSTRWK Format (p. 239)</td>
<td>Writes a numeric value as the day-of-the-week in the specified locale.</td>
</tr>
<tr>
<td>YEN Format (p. 271)</td>
<td>Writes numeric values with yen signs, commas, and decimal points.</td>
</tr>
</tbody>
</table>

### Dictionary

**B8601DX Format**

Adjusts a Coordinated Universal Time (UTC) datetime value to the user 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
  ISO 8601
- **Alignment:** Left
- **Supports:** ISO 8601 Element 5.4.1, complete representation
Syntax

B8601DXₜₜ

Syntax Description

ₜₜ
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:

• 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

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.
Statement | Result
--- | ---
data _null_;
t='01Feb2013T12:34:56'dt ;
put t b8601dx.;
run;

options timezone='America/Adak';
data _null_;
t='01Feb2013T12:34:56'dt ;
put t b8601dx.;
run;

20130201T073456-0500

20130201T023456-1000

See Also

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

**B8601LX 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

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

**Syntax**

*B8601LX* `w`

**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.
**Example**

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

```sas
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 Viya Formats and Informats: Reference*

---

**B8601TX Format**

Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 basic time notation `hhmmss+|-hhmm`.

**Categories:** Date and Time  
 ISO 8601  

**Alignment:** Right  

**Supports:** ISO 8601 Elements 5.3.3 and 5.3.4
**Syntax**

B8601TX\(w\).

**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 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 datetime values by using the following ISO 8601 basic time notation:

- \(hh:mm:ss\)\(+|–hh:mm\)

  \(hh\)

  - 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, +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.
Statement | Result
---|---
data _null_;
t='12:34:56't;
put t b8601tx.;
run; | 073456-0500

options timezone='America/Adak';
data _null_;
t='12:34:56't;
put t b8601tx.;
run; | 023456-1000

See Also

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

### BESTDOTX Format

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

**Category:** Numeric

**Alignment:** Right

### Syntax

**BESTDOTX**\(w\).

### 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 -0.01 exclusively, use a field width of at least 8.

### Details

If the NLDECESEPARATOR system option is disabled, the BEST\(w\) and BESTDOTX\(w\) formats process data the same way. If the NLDECESEPARATOR 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 BESTw.</th>
<th>Separator character used by BESTDOTXw.</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 NLDECSEPARATOR option enabled and the locale set to fr_FR.

```sas
options nld locale=fr_FR;
da_null_
  x=1.2;
  call symput('macx', put(x, BESTDOTX.));
  put x; /* Result is printed as "1,2" */
y=put(x, BESTDOTX.); /* Result is printed as "1.2" */
run;
&put &macx;
run;
```

**See Also**

- “BEST Format” in *SAS Viya Formats and Informats: Reference*
- “NLDECSEPARATOR System Option” on page 584

**$CPTDW 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
Syntax
$CPTDWw.$

Syntax Description

$w$

specifies the width of the output field.

Default 200

Range 1–32767

Comparisons

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>$118</td>
</tr>
</tbody>
</table>

See Also

Format:

- “$CPTWD Format” on page 89

Informat:

- “$CPTDW Informat” on page 442
- “$CPTWD Informat” on page 443

$CPTWD 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

Syntax

$SCPTWDw.$
**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 “이다”.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $cptwd3.;</td>
<td>$[]</td>
</tr>
</tbody>
</table>

**See Also**

- “\( $CPTDW \) Format” on page 88
- “\( $CPTDW \) Informat” on page 442
- “\( $CPTWD \) Informat” on page 443

---

**$EBCDIC Format**

Converts native format character data to EBCDIC representation.

- **Category**: Character
- **Alignment**: Left
- **Note**: UTF-8 is the only supported session encoding. $EBCDIC format works only for 7bit ASCII characters.

**Syntax**

\( SEBCDICw \)
Syntax Description

`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 LOCALE= option that is specified when SAS starts. For example, if the locale was set to en_US locale, the default encoding that is used by the $EBCDICw. format is Open ed-1047. If the locale is 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 17.2 on page 636.

You can specify the translation table that is used to map characters between EBCDIC and ASCII by using the MAPEBCDIC2ASCII system option. For more information, see “MAPEBCDIC2ASCII= System Option” on page 583.

Comparisons

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

Example

```sas
put name $ebcdic3.;
```

<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 two hexadecimal characters correspond to one byte of binary data, and each byte corresponds to one character.

---

E8601DX Format

Adjusts a Coordinated Universal Time (UTC) datetime value to the user 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
Synt<136>ex

\texttt{E8601DXw}.

\textbf{Syntax Description}

\texttt{w}

specifies the width of the output field.

\begin{itemize}
  \item \texttt{Default} \hspace{1cm} 26
  \item \texttt{Range} \hspace{1cm} 20–35
\end{itemize}

\textbf{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 \texttt{TIMEZONE=} system option. If the \texttt{TIMEZONE=} option is not set, the user local date and time are based on the local date and time. The \texttt{E8601DX} format writes SAS datetime values by using the following ISO 8601 basic datetime notation:

- \texttt{yyyy-mm-ddThh:mm:ss+hh:mm}

  \begin{itemize}
    \item \textit{yyyy} is a four-digit year.
    \item \textit{mm} is a two-digit month (zero padded) between 01 and 12.
    \item \textit{dd} is a two-digit day of the month (zero padded) between 01 and 31.
    \item \textit{hh} is a two-digit hour (zero padded) between 00 and 23.
    \item \textit{mm} is a two-digit minute (zero padded) between 00 and 59.
    \item \textit{ss} is a two-digit second (zero padded) between 00 and 59.
    \item \texttt{+|−hh:mm} is an hour and minute signed offset from zero meridian time. The offset must be \texttt{+|−hh:mm} (that is, \texttt{+} or \texttt{−} and four characters).
  \end{itemize}

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

\textbf{Restriction:} The shorter form \texttt{+|−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> ;</td>
<td>2013-02-01T07:34:56-05:00</td>
</tr>
<tr>
<td>t='01Feb2013T12:34:56'dt ;</td>
<td></td>
</tr>
<tr>
<td>put t e8601dx.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options timezone='America/Adak';</td>
<td>2013-02-01T02:34:56-10:00</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>t='01Feb2013T12:34:56'dt ;</td>
<td></td>
</tr>
<tr>
<td>put t e8601dx.;</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 Viya Formats and Informats: Reference

E8601LX Format

Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 extended notation `yyyy-mm-ddThh:mm:ss±|–hh:mm`.

**Categories:** Date and Time  
ISO 8601

**Alignment:** Right

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

**Syntax**

```
E8601LXw
```

**Syntax Description**

`w`

specifies the width of the output field.

**Default** 26

**Range** 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 the 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**

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

```
blx='01Feb2013T12:34:56'dt;
pblx e8601lx.;
```

<table>
<thead>
<tr>
<th>Value of blx</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1675341296</td>
<td>2013-02-01T12:34:56-05:00</td>
</tr>
</tbody>
</table>

**See Also**

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

---

**E8601TX Format**

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

**Categories:** Date and Time

**Alignment:** Right

**Supports:** ISO 8601 Elements 5.3.3 and 5.3.4
Syntax

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 the following ISO 8601 basic time notation:

- \( hh:mm:ss\)\(\pm\)\( hh:mm\)

  - \( hh\) 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.
  - \(\pm\)\( hh:mm\) is an hour and minute signed offset from zero meridian time. The offset must be \(\pm hh:mm\) (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, +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 \(\pm hh\) 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't; 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't; 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 Viya Formats and Informats: Reference*

---

**EURO Format**

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

- **Categories:** Numeric, CAS
- **Alignment:** Right

**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 EURO\textit{w.d} format is similar to the EUROX\textit{w.d} format, but EUROX\textit{w.d} format reverses the roles of the decimal point and the comma. This convention is common in European countries.

- The EURO\textit{w.d} format is similar to the DOLLAR\textit{w.d} format, except that DOLLAR\textit{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 euro10.2;</td>
<td>E1,254.71</td>
</tr>
<tr>
<td>put amount euro5.;</td>
<td>E1255</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  
E1255  
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;
   put x euro10.;
   put x euro10.2;
   put x euro10.4;
   put x euro11.;
   put x euro11.3;
   put x euro12.;
   put x euro12.2;
   put x euro13.;
   put x euro13.2;
   datalines;
      333
      4444
      55555
      666666
      7777777
      88888888
      999999999
      1234561234
;run;

See Also

Format:
- “EUROX Format” on page 99

Informats:
- “EURO Informat” on page 444
- “EUROX Informat” on page 445
EUROX 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.

Categories: Numeric
CAS
Alignment: Right

Syntax

EUROX\[w.d\]

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 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>Statements</td>
<td>Results</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>put amount eurox5.;</td>
<td>€1255</td>
</tr>
<tr>
<td>put amount eurox9.2;</td>
<td>€1.254,71</td>
</tr>
<tr>
<td>put amount eurox15.3;</td>
<td>€1.254,710</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:  
  €1.254,71  
  €1255  
  €1.254,71  
  €1.254,710  

  /* This code determines the default length. */  
data _null_;  
  input x;  
  put x eurox.;  
  datalines;  
1  
  22  
  333  
  4444  
  5555  
  666666  
  7777777  
  88888888  
  999999999  
  1234561234  
;run;  
SAS Log:  
  €1  
  €22  
  €333  
  €4.444  
  €5.555  
  €666666  
  €7.78E6  
  €8.89E7  
  €1B9  
  €1.23B9  
```

*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.
MINGUO Format

Writes date values as Taiwanese dates in the form yyyyymmdd.

Category: Date and Time
Alignment: Left

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>(x=\text{put}(12054,\text{minguo7.});) put (x=;)</td>
<td>(x=820101)</td>
</tr>
<tr>
<td>(x=\text{put}(12054,\text{minguo9.});) put (x=;)</td>
<td>(x=82/01/01)</td>
</tr>
<tr>
<td>(x=\text{put}(12054,\text{minguo10.});) put (x=;)</td>
<td>(x=0082/01/01)</td>
</tr>
<tr>
<td>(x=\text{put}(18993,\text{minguo7.});) put (x=;)</td>
<td>(x=1000101)</td>
</tr>
<tr>
<td>(x=\text{put}(18993,\text{minguo9.});) put (x=;)</td>
<td>(x=100/01/01)</td>
</tr>
<tr>
<td>(x=\text{put}(18993,\text{minguo10.});) put (x=;)</td>
<td>(x=0101/01/01)</td>
</tr>
<tr>
<td>(x=\text{put}(-20088,\text{minguo7.});) put (x=;)</td>
<td>(**********)</td>
</tr>
<tr>
<td>(x=\text{put}(-20088,\text{minguo9.});) put (x=;)</td>
<td>(**********)</td>
</tr>
<tr>
<td>(x=\text{put}(-20088,\text{minguo10.});) put (x=;)</td>
<td>(**********)</td>
</tr>
</tbody>
</table>

**See Also**

**Informat:**
- “MINGUO Informat” on page 447

**NENGO Format**

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

**Categories:** Date and Time

CAS

**Alignment:** Left

**Syntax**

\texttt{NENGOw};
**Syntax Description**

\( w \)

specifies the width of the output field.

Default 10

Range 2–10

**Details**

The NENGO\( w \). format writes SAS date values in the form \( e.yymmdd \), where

- \( e \) is the first letter of the name of the emperor (Meiji, Taisho, Showa, or Heisei).
- \( yy \) 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.

```sas
data _null_
  date=15342;
  put date nengo3.;
  put date nengo6.;
  put date nengo8.;
  put date nengo9.;
  put date nengo10.;
run
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date nengo3.;</td>
<td>H14</td>
</tr>
<tr>
<td>put date nengo6.;</td>
<td>H14/01</td>
</tr>
<tr>
<td>put date nengo8.;</td>
<td>H.140102</td>
</tr>
<tr>
<td>put date nengo9.;</td>
<td>H14/01/02</td>
</tr>
<tr>
<td>put date nengo10.;</td>
<td>H.14/01/02</td>
</tr>
</tbody>
</table>
See Also

Informat:

- “NENG0 Informat” on page 449

NLBEST Format

Writes the best numerical notation based on the locale.

Categories: Numeric
CAS
Alignment: Right

Syntax

NLBEST <w>

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, 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 Viya Formats and Informats: Reference.

Example

The following code produces results based on the locale:

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.;

<table>
<thead>
<tr>
<th>Locales</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>locale=English_UnitedStates</td>
<td>-126E4</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>===========</td>
</tr>
<tr>
<td></td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>-.1</td>
</tr>
<tr>
<td></td>
<td>===========</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>===========</td>
</tr>
<tr>
<td></td>
<td>1.26E6</td>
</tr>
<tr>
<td></td>
<td>1E6</td>
</tr>
</tbody>
</table>

| locale=German_Germany | -126E4        |
|                       | ***           |
|                       | ===========   |
|                       | -0,1          |
|                       | -.1           |
|                       | ===========   |
|                       | 0,1           |
|                       | 0,1           |
|                       | ===========   |
|                       | 1,26E6        |
|                       | 1E6           |

| locale=ar_BH         | 126E4-        |
|                     | ***           |
|                     | ===========   |
|                     | 0,1-          |
|                     | .1-           |
|                     | ===========   |
|                     | 0.1           |
|                     | 0.1           |
|                     | ===========   |
|                     | 1.26E6        |
|                     | 1E6           |
```
**NLDATE Format**

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

**Categories:** Date and Time
CAS

**Alignment:** Left

**Note:** Use the NLDATE format in the CAS server instead of the WORDDATE format.

**Syntax**

\[ \text{NLDATE}_w. \]

**Syntax Description**

\( w \)

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

**Note:** The format pattern, long, medium, short, is used according to the specified width.

**Default** 20

**Range** 10–200

**Comparisons**

NLDATE\( w \). is similar to DATE\( w \). and WORDDATE\( w \). except that NLDATE\( w \). is locale-specific.

**Example**

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+----1----+----2</td>
<td></td>
</tr>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>February 24, 2003</td>
</tr>
<tr>
<td>put day nldate.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>24. Februar 2003</td>
</tr>
<tr>
<td>put day nldate.;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:
- “NLDATEMN Format” on page 112
- “NLDATEW Format” on page 114
- “NLDATEWN Format” on page 116

NLDATEL Format

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, date, month, year.

Categories: Date and Time
CAS

Alignment: Left

Syntax

NLDATELw.

Syntax Description

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

Default 18

Range 2–220

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.
NLDATEM Format

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

<table>
<thead>
<tr>
<th>Categories:</th>
<th>Date and Time</th>
<th>CAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

NLDATEM\texttt{w}

Syntax Description

\texttt{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>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>2–200</td>
</tr>
</tbody>
</table>

Details

NLDATEM writes the date in a medium-uniform pattern such as Nov 19, 2012.

Example

This example uses the date November 19, 2012.

Statements  | Results |
-------------|---------|
\begin{verbatim}
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATEL min=2 default=18 max=200 +++-
               November 19, 2012
               11/19/2012
               Nov 19, 2012
               November 19, 2012
               November 19, 2012
               November 19, 2012
               November 19, 2012
               November 19, 2012
               November 19, 2012
              
run;
\end{verbatim}
Statements

```sas
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATEM min=2 default=14 max=200 ---+
   Nov 19, 2012
   11/19/12
   Nov 19, 2012
   Nov 19, 2012
";  
put dy nldatem.;  
put dy nldatem8.;  
put dy nldatem14.;  
put dy nldatem200.;  
rung;
```

### NLDATEMD 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:** Date and Time

**CAS**

**Alignment:** Left

### Syntax

```
NLDATEMDw.
```

**Syntax Description**

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

**Default** 16

**Range** 6-200

### Example

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put 1 nldatemd.;</td>
<td>January 02</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLDATEYM Format” on page 117
NLDATEMDL 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:** Date and Time
CAS

**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 full length of the month and the day, such as November 19.

**Example**

The following example uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+---- NLDATEMDL min=5 default=12 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 19</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19</td>
</tr>
<tr>
<td>put &quot;<em>---- NLDATEMDL min=5 default=12 max=200 ----</em>&quot;;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dy nldatemdl.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dy nldatemdl15.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dy nldatemdl19.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dy nldatemdl112.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dy nldatemdl1200.;</td>
<td>November 19</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
**NLDATEMDM 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:** Date and Time  
CAS  

**Alignment:** Left

**Syntax**

NLDATEMDM <w>.

**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 uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATEMDM min=5 default=9 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Nov 19</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19</td>
</tr>
<tr>
<td>put &quot;+--- NLDATEMDM min=5 default=9 max=200 ----&quot;;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dy nldatemdm.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dy nldatemdm5.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dy nldatemdm9.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dy nldatemdm200.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

---

**NLDATEMDS 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.
Syntax

**NLDATEMDS**

**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 uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+---- NLDATEMDS min=5 default=5 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>11/19</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19</td>
</tr>
<tr>
<td>put &quot;----- NLDATEMDS min=5 default=5 max=200 ----&quot;;</td>
<td>11/19</td>
</tr>
<tr>
<td>max=200 ++&quot;;</td>
<td>11/19</td>
</tr>
<tr>
<td>put dy nldatemds.;</td>
<td></td>
</tr>
<tr>
<td>put dy nldatemds5.;</td>
<td></td>
</tr>
<tr>
<td>put dy nldatemds5.;</td>
<td></td>
</tr>
<tr>
<td>put dy nldatemds200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLDATEMN 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**: Date and Time

**CAS**

**Alignment**: Left
Syntax

NLDATEMNw.

Syntax Description

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

<table>
<thead>
<tr>
<th>Default</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>4–200</td>
</tr>
</tbody>
</table>

Comparisons

NLDATEMNw. is similar to MONNAMEw. except that NLDATEMNw. is locale-specific.

Example

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; put month nldatemn.;</td>
<td>February</td>
</tr>
<tr>
<td>options locale=German_Germany; put month nldatemn.;</td>
<td>Februar</td>
</tr>
</tbody>
</table>

See Also

Formats:

- “NLDATE Format” on page 106
- “NLDATEW Format” on page 114
- “NLDATEWN Format” on page 116

NLDATES 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: Date and Time
CAS
**Syntax**

NLDATES\(w\).

**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 uses the date November 19, 2012.

---

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATES min=2 default=10 max=200 11/19/2012</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>11/19/12</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19/12</td>
</tr>
<tr>
<td>put &quot;+--- NLDATES min=2 default=10 max=200 ++++&quot;;</td>
<td>11/19/2012</td>
</tr>
<tr>
<td>put dy nldates.;</td>
<td>11/19/2012</td>
</tr>
<tr>
<td>put dy nldates8.;</td>
<td></td>
</tr>
<tr>
<td>put dy nldates10.;</td>
<td></td>
</tr>
<tr>
<td>put dy nldates200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

---

**NLDATEW Format**

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:** Date and Time  
CAS

**Alignment:** Left

**Note:** Use the NLDATEW format in the CAS server instead of the WEEKDATE format.
Syntax

NLDATEWw.

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 60 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>--+----1----+----2</td>
<td></td>
</tr>
<tr>
<td>options locale=English_UnitedStates; date=15760; x=put(date,nldatew.); y=put(date,nldatew20.); z=put(date,nldatew200.); run;</td>
<td>Monday, February 24, 2003 Monday, February 24, 2003 Monday, February 24, 2003</td>
</tr>
<tr>
<td>options locale=German_Germany; date=15760; x=put(date,nldatew.); y=put(date,nldatew20.); z=put(date,nldatew200.); run;</td>
<td>Mo., 24. Feb 2003 Mo., 24. Feb 2003 Montag, 24. Februar 2003</td>
</tr>
</tbody>
</table>
See Also

Formats:
- “NLDATE Format” on page 106
- “NLDatemn Format” on page 112
- “NLDATEWN Format” on page 116

NLDATEWN 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.

<table>
<thead>
<tr>
<th>Categories:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td></td>
</tr>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Default</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>4–200</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>Monday</td>
</tr>
<tr>
<td>put date nldatewn;</td>
<td></td>
</tr>
</tbody>
</table>
### NLDATEYM 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:
- Date and Time
- CAS

#### 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;

data _null_;
  format d nldateym6.;
  d = '17OCT14'd;
  put d=;
```

---

**Statements** | **Results**
--- | ---
options locale=German_Germany; | Montag
put date nldatewn; |
Example
This example uses the spanish_Spain locale option, and the date of August 2010.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=spanish_Spain;</td>
<td></td>
</tr>
<tr>
<td>data_null_;</td>
<td></td>
</tr>
<tr>
<td>dy=today();</td>
<td>agosto de 2010</td>
</tr>
<tr>
<td>x=put(dy, nldateym.);</td>
<td>ago de 10</td>
</tr>
<tr>
<td>y=put(dy, nldateym12.);</td>
<td>agosto de 2010</td>
</tr>
<tr>
<td>z=put(dy, nldateym200.);</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLDATEMD Format” on page 109

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

**Categories:** Date and Time

**CAS:**

**Alignment:** Left

**Syntax**

NLDATEYML\textsubscript{w}.

**Syntax Description**

\textsubscript{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

NLDATEYML writes the date in a long-uniform pattern with abbreviations for the month and year, such as November 2012.

Example

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATEYML min=5 default=14 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/2012</td>
</tr>
<tr>
<td>put &quot;++++ NLDATEYML min=5 default=14 max=200 ---+&quot;;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dy nldateyml.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml7.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml11.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml14.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATEYMM 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.

<table>
<thead>
<tr>
<th>Categories:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td></td>
</tr>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

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 Nov 2012.
Example

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+++- NLDATEYMS min=5 default=11 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/2012</td>
</tr>
<tr>
<td>put &quot;+++- NLDATEYMS min=5 default=11 max=200 ---+&quot;;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dy nldateymm.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dy nldateyymm7.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dy nldateyymm11.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dy nldateyymm200.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATEYMS 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:** Date and Time
- **CAS:**
- **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 uses the date November 19, 2012.
NLDATEYQ 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:** Date and Time
CAS

**Alignment:** Left

**Syntax**

\[ \text{NLDATEYQ}w \]

**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 60 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

**Example**

This example uses the fr_FR locale option.
Statements

options locale=fr_FR;

data _null_;

dy=today();
dt=datetime();

put "+--- NLDATEYQ min=4 default=16 max=200 ---+
   16 T3 08
   4 ****
   14 T3 08
   32 3e trimestre 2008
   200
   3e trimestre 2008
";

run;

Results

+--- NLDATEYQ min=4 default=16 max=200 ---+
   16 T3 08
   4 ****
   14 T3 08
   32 3e trimestre 2008
   200
   3e trimestre 2008

--- NLDATEYQ 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:
- Date and Time
- CAS

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 4th quarter 2012.

Example

The following example uses the date November 19, 2012.
NLDATEYQM 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:** Date and Time

**CAS:**

**Alignment:** Left

### Syntax

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 Q4 2012.

### Example

The following example uses the date November 19, 2012.

```sas
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;
```
### NLDATEYQS 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:** Date and Time  
CAS  
**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.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4–200</td>
</tr>
</tbody>
</table>

### Details

NLDATEYQS writes the date in a short-uniform pattern with numbers and delimiters for the year and year’s quarter value, such as 2012/4.

### Example

The following example uses the date November 19, 2012.
Statements | Results
---|---
data _null_; | +++- NLDATEYQS min=6 default=6 max=200 +++
dt = datetime(); | 2012/4
dy = date(); | 12/4
put "+++ NLDATEYQS min=4 default=6 max=200 +++";
 | 2012/4
 | 2012/4
put dy nldateyqs.;
 | 2012/4
put dy nldateyqs4.;
 | 2012/4
put dy nldateyqs6.;
 | 2012/4
put dy nldateyqs200.;
run;

**NLDATEYR Format**

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

**Categories:** Date and Time  
CAS  
**Alignment:** Left

**Syntax**

NLDATEYRw.

**Syntax Description**

w

specifies the width of the output field.

**Default**

16

**Range**

2–200

**Example**

This example uses the fr_FR locale option.
NLDATEYW 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:** Date and Time  
CAS

**Alignment:** Left

---

**Syntax**

NLDATEYW\_w.

**Syntax Description**

\_w  
specifies the width of the output field.

**Default** 16  
**Range** 5–200

---

**Example**

This example uses the fr_FR locale option.
### NLDATM Format

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

#### Categories:
- Date and Time
- CAS

#### Alignment:
- Left

#### Syntax

```plaintext
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

#### Comparisons

The NLDATMw. format is similar to the DATETIMEw. format except that the NLDATMw. 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.
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>24Feb2003:12:39:43</td>
</tr>
<tr>
<td>put day nldatm.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>24. Februar 2003 12.39 Uhr</td>
</tr>
<tr>
<td>put day nldatm.;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Formats:**
- “NLDATMAP Format” on page 128
- “NLDATMTM Format” on page 137
- “NLDATMW Format” on page 139

### NLDATMAP 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:**  
Date and Time  
CAS

**Alignment:**  
Left

**Syntax**

NLDATMAP\_w.

**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 NLDATMAP\_w. format is similar to DATEAMP\_M\_w. except that the NLDATMAP\_w. format is locale-specific.
Example

These examples use the input value of 1361709583, which is the SAS date-time value that corresponds to 12:39:43 p.m. on February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>February 24, 2003 12:39:43 PM</td>
</tr>
<tr>
<td>put event nldatmap.;</td>
<td></td>
</tr>
<tr>
<td>options locale=Spanish_Mexico;</td>
<td>24/02/2003 12:39:43 PM</td>
</tr>
<tr>
<td>put event nldatmap.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “NLDATM Format” on page 127
- “NLDATMTM Format” on page 137
- “NLDATMW Format” on page 139

NLDATMDT 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: Date and Time
            CAS
Alignment:  Left

Syntax

NLDATMDT\(w\).

Syntax Description

\(w\)

specifies the width of the output field

Default 18

Range 10-200

Example

This example uses the en_US locale option.
Statements

```sas
options locale=en_US;
data _null_
    x=86400;
    put x nldatmdt.;
run;
options locale=en_US;
data _null_
    x=86400;
    put x dtdate.;
run;
```

**Results**

January 02, 1960

02JAN60

---

## See Also

**Formats:**

- "NLDATMMD Format" on page 131

---

### NLDATMML Format

Converts a SAS datetime 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.

**Categories:**

- Date and Time
- CAS

**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.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>9–200</td>
</tr>
</tbody>
</table>

**Details**

NLDATMML writes the date in a long-uniform pattern with the full length of the month date, year, and time, such as, November 19, 2012 02:57:44 PM.

**Example**

This example uses the date November 19, 2012.
Statements

data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATML min=9 default=30 max=200 ---+
November 19, 2012 02:57:44 PM
11/19/2012 14:57:44
Nov 19, 2012 02:57:44 PM
November 19, 2012 02:57:44 PM
November 19, 2012 02:57:44 PM
run;

Results

+NLDATMM Format

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.

<table>
<thead>
<tr>
<th>Categories:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td></td>
</tr>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

\texttt{NLDATMM}_w.

Syntax Description

\texttt{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>24</th>
</tr>
</thead>
</table>

| Range | 9–200 |

Details

NLDATMM writes the date in a medium-uniform pattern with abbreviations of the month and week names, such as Nov 19, 2012 02:51:40 PM.

+NLDATMMD 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.

<table>
<thead>
<tr>
<th>Categories:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td></td>
</tr>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>
Syntax

\texttt{NLDATMMD}_{w}\texttt{.}

\textbf{Syntax Description}

\texttt{w}

specifies the width of the output field.

Default \hspace{0.5cm} \texttt{16}

Range \hspace{0.5cm} \texttt{6–200}

\textbf{Example}

This example uses the \texttt{en\_US} locale option.

\begin{verbatim}
options locale=en_US;
data _null_
x=put(86400,nldatmmd.);
put x=;
run;
\end{verbatim}

\begin{tabular}{l|l}
\hline
\textbf{Statement} & \textbf{Result} \\
\hline
options locale=en_US;
data _null_
x=put(86400,nldatmmd.);
put x=;
run; & January 02 \\
\hline
\end{tabular}

\textbf{See Also}

\textbf{Format:}

\begin{itemize}
\item “NLDATMYM Format” on page 142
\end{itemize}

\textbf{NLDATMMDL 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.

\textbf{Categories:} Date and Time

\textbf{CAS}

\textbf{Alignment:} Left

\textbf{Syntax}

\texttt{NLDATMMDL}_{w}\texttt{.}
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 November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+---- NLDATMMDL min=5 default=12 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 19</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19</td>
</tr>
<tr>
<td>put &quot;----- NLDATMMDL min=5 default=12 max=200 -----&quot;;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dt nldatmmdl.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl5.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl9.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl12.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl200.;</td>
<td>November 19</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMMDM 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.

Categories: Date and Time

CAS

Alignment: Left

Syntax

NLDATMMDM\(w\).

Syntax Description

\(w\)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.
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 November 19, 2012.

```
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;  
```

--- NLDATMMDM min=5 default=9 max=200 ---
Nov 19
11/19
Nov 19
Nov 19

NLDATMMDS 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.

**Categories:** Date and Time

**CAS:**

**Alignment:** Left

**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 11/19.

Example

The following example uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMMDS min=5 default=5 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>11/19</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMMDS min=5 default=5 max=200 ++++&quot;;</td>
<td>11/19</td>
</tr>
<tr>
<td>max=200 ++-+&quot;;</td>
<td>11/19</td>
</tr>
<tr>
<td>put dt nldatmmds.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmmds5.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmmds5.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmmds200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMMN 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:** Date and Time, CAS
- **Alignment:** Left

**Syntax**

NLDATMMN\textsubscript{\(w\)}.

**Syntax Description**

\(w\)

- specifies the width of the output field.

- **Default:** 9
- **Range:** 4–200

**Example**

This example uses the en_US locale option.
### NLDATMS Format

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.

<table>
<thead>
<tr>
<th>Categories</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td></td>
</tr>
<tr>
<td>Alignment</td>
<td>Left</td>
</tr>
</tbody>
</table>

#### Syntax

NLDATMS\(w\).

#### 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>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>9–200</td>
</tr>
</tbody>
</table>

#### 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 November 19, 2012.

---

```
data _null_;  +--- NLDATMMN min=4 default=10
   dt = datetime();       max=200 ---+
   dy = date();          October
   put "+--- NLDATEMN min=4 default=10
   ";                     max=200 ---+
   put dt nldatmmn.;      October
   put dt nldatmmn4.;     October
   put dt nldatmmn10.;    October
   put dt nldatmmn200.;   October
   run;                   
```

---
Statements

data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATMS min=9 default=19 max=200 ---+
   11/19/2012 14:38:52
   11/19/2012 14:38:52
   11/19/2012 14:38:52
   11/19/2012 14:38:52";
   put dt nldatms.;  
   put dt nldatms10.;  
   put dt nldatms19.;  
   put dt nldatms200.;  
runt;

Results

NLDATMTM Format

Converting 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: Date and Time
CAS
Alignment: Left

Syntax

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.

Statements

Results

----+----1
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>12:39:43</td>
</tr>
<tr>
<td>put event nldatmtm.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>12.39 Uhr</td>
</tr>
<tr>
<td>put event nldatmtm.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “NLDATM Format” on page 127
- “NLDATMAP Format” on page 128
- “NLDATMW Format” on page 139

NLDATMTZ Format

Converts the time portion of the SAS datetime of the locale to the time of day and time zone.

Categories: Date and Time
CAS
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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td>x=10 h 40 -0400</td>
</tr>
<tr>
<td>data test;</td>
<td></td>
</tr>
<tr>
<td>x=datetime();</td>
<td></td>
</tr>
<tr>
<td>put x=nldatmtz.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
NLDATMW Format

Converts SAS datetime values to the locale sensitive datetime string as the day of the week and the datetime.

**Categories:** Date and Time
CASS

**Alignment:** Left

**Syntax**

\textbf{NLDATMW}_w$
$

**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 60 for information about recommended widths for locale and encoding combinations. You might need to use the suggested width for the NLDATMW format.

**Comparisons**

The NLDATMW\textsubscript{w} format is similar to the TWMDY\textsubscript{w} format except that the NLDATMW\textsubscript{w} 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+----1---+----2---+----3---+</td>
<td></td>
</tr>
</tbody>
</table>
### See Also

**Formats:**
- “NLDATM Format” on page 127
- “NLDATMAP Format” on page 128
- “NLDATMTM Format” on page 137

### NLDATMWN 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:** Date and Time  
CAS

**Alignment:** Left

#### Syntax

NLDATMWNw.

#### Syntax Description

w  
specifies the width of the output field.

**Default** 9  
**Range** 4–200
Example

This example writes the SAS datetime value as a day of the week.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMWN min=4 default=9 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Monday</td>
</tr>
<tr>
<td>dy = date();</td>
<td>Mon</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMWN min=4 default=9 max=200 ---+&quot;;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn.;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn4.;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn9.;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn200.;</td>
<td>Monday</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMWZ Format

Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.

**Categories:** Date and Time

**CAS:**

**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 60 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.
Statements | Result
---|---
```plaintext
options locale=fr_FR;
data test;
x=datetime();
put x=nldatmwz.;
run;
```
x=vendredi 18 mars 2011 10 h 40 -0400

### NLDATMYM Format
Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the month.

**Categories:** Date and Time

**Alignment:** Left

**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.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
</table>
| ```plaintext
options locale=en_US;
data _null_;  
x=put(86400,nldatmym.);
y=put(86400,nldatmym12.);
put x=;
pay y=;
run;
```
| January 1960 | January 1960 |
See Also

Format:

• “NLDATMMD Format” on page 131

NLDATMYML 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: Date and Time
CAS

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 November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMYML min=5 default=14 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/2012</td>
</tr>
<tr>
<td>put &quot;*--- NLDATMYML min=5 default=14 max=200 +++\n</td>
<td>max=200 +++\n</td>
</tr>
<tr>
<td>put dt nldatmyml7.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dt nldatmyml11.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyml14.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyml200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
NLDATMYMM 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:** Date and Time

**CAS**

**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.

**Default** 11

**Range** 5–200

**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 November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+----- NLDATMYMM min=5 default=11 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/2012</td>
</tr>
<tr>
<td>put &quot;+----- NLDATMYMM min=5 default=11 max=200 ----&quot;;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymm.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymm7.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymm11.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymm200.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYMS Format

Converts a SAS date value to the date string of the specified locale, and then writes the month and year with numbers and delimiters.
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 delimiters for the month and year, such as 11/2012.

Example

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMYMS min=5 default=7 max=200 ++++</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>11/2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/12</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMYMS min=5 default=7 max=200 ++++&quot;;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dt nldatmyms.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dt nldatmyms5.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dt nldatmyms7.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dt nldatmyms200.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYQ Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.
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 60 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Example

This example uses the fr_FR locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td>+--- NLDATMYQ min=4 default=16 max=200 ---+</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>16 T3 08</td>
</tr>
<tr>
<td>dy=today();</td>
<td>4 ****</td>
</tr>
<tr>
<td>dt=datetime();</td>
<td>14 T3 08</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMYQ min=4 default=16 max=200 ---+&quot;;</td>
<td>32 3e trimestre 2008</td>
</tr>
<tr>
<td>put '16' +5 dt nldatmyq.;</td>
<td>200 3e trimestre 2008</td>
</tr>
<tr>
<td>put '4' +5 dt nldatmyq4.;</td>
<td></td>
</tr>
<tr>
<td>put '14' +5 dt nldatmyq14.;</td>
<td></td>
</tr>
<tr>
<td>put '32' +5 dt nldatmyq32.;</td>
<td></td>
</tr>
<tr>
<td>put '200' +5 dt nldatmyq200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYQL 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.

Categories: Date and Time

CAS

Alignment: Left
Syntax

NLDATMYQ\textsubscript{w}.

**Syntax Description**

\texttt{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 November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMYQL min=4 default=18 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>2012/4</td>
</tr>
<tr>
<td>put &quot;+++ NLDATMYQL min=4 default=18 max=200 +++&quot;;</td>
<td>Q4 2012</td>
</tr>
<tr>
<td>put dt nldatmyql.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>put dt nldatmyql6.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>put dt nldatmyql7.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>put dt nldatmyql18.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>put dt nldatmyql200.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLDATMYQM 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:** Date and Time, CAS
- **Alignment:** Left

**Syntax**

NLDATMYQM\textsubscript{w}.
Syntax Description

\[ w \]

\[ w \]
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 Q4 2012.

Example

The following example uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMYQM min=4 default=7 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Q4 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>2012/Q4</td>
</tr>
<tr>
<td>put &quot;*---- NLDATMYQM min=4 default=7 max=200 ---&quot;;</td>
<td>Q4 2012</td>
</tr>
<tr>
<td>max=200 ----&quot;;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyqm.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyqm6.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyqm7.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyqm200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

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 delimiters.

Categories: Date and Time

CAS

Alignment: Left

Syntax

NLDATMYQS \[ w \].

Syntax Description

\[ w \]
specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.
Details

NLDATMYQS writes the date in a short-uniform pattern of the year and year’s quarter value using numbers and delimiters , such as 2012/4.

Example

The following example uses the date November 19, 2012.

```
 Statements                      Results
  data _null_;                   +---- NLDATMYQS min=4 default=6 max=200 ----+
  dt = datetime();               2012/4
  dy = date();                   12/4
  put "+---- NLDATMYQS min=4 default=6 max=200 ----";    2012/4
    put dt nldatmyqs.;           2012/4
    put dt nldatmyqs4.;          2012/4
    put dt nldatmyqs6.;          2012/4
    put dt nldatmyqs200.;        2012/4
  run;
```

NLDATMYR Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.

**Categories:** Date and Time

**CAS:**

**Alignment:** Left

**Syntax**

```
NLDATMYR w.
```

**Syntax Description**

`w`

specifies the width of the output field.

**Default:** 16

**Range:** 2–200
Example

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td>+--- NLDATMYR min=2 default=16 max=200 ---+</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>dy=today();</td>
<td>2008</td>
</tr>
<tr>
<td>dt=datetime();</td>
<td></td>
</tr>
<tr>
<td>put &quot;+--- NLDATMYR min=2 default=16 max=200 ---+&quot;;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyr.;</td>
<td>08</td>
</tr>
<tr>
<td>put dt nldatmyr2.;</td>
<td>2008</td>
</tr>
<tr>
<td>put dt nldatmyr32.;</td>
<td>2008</td>
</tr>
<tr>
<td>put dt nldatmyr200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYW Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.

**Categories:** Date and Time
CAS

**Alignment:** Left

**Syntax**

NLDATMYWw.

**Syntax Description**

w

specifies the width of the output field.

Default 16

Range 5–200

**Example**

This example uses the fr_FR locale option.
Statements

```sas
options locale=fr_FR;

data _null_;
dy=today();
dt=datetime();
put "+--- NLDATMYW min=5 default=16
max=200 ---+
16 Week 33 2008
5 *****
8 W33 08
32 Week 33 2008
200
Week 33 2008
+--- NLDATMYWW min=5 default=16
max=200 ---+
16 Week 33 2008
5 *****
8 W33 08
32 Week 33 2008
200
Week 33 2008
```

Results

```
+--- NLDATMYW min=5 default=16
max=200 ---+
16 Week 33 2008
5 *****
8 W33 08
32 Week 33 2008
200
Week 33 2008
```

---

**NLDATMZ Format**

Converts SAS datetime values to the locale-sensitive datetime string as time zone and datetime.

**Categories:** Date and Time  
CAS

**Alignment:** Left

---

**Syntax**

`NLDATMZw.`

**Syntax Description**

`w`  
specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>16–200</td>
</tr>
</tbody>
</table>

**Example**

This example uses the current datetime value.
Statements | Result
---|---
```options locale=fr_FR;
data test;
x=datetime();
put x=nldatmz.;
run;```
```x=18 mars 2011 10 h 40 -0400```

### NLMNIAED Format

Writes the monetary format of the international expression for the United Arab Emirates.

**Category:** Numeric

**Alignment:** Left

### Syntax

```
NLMNIAED 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: 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>
</table>
| ```put x=;```
| (AED1,234.57) |
| ```put y=;```
| $-1,234.57 |
See Also

Format:
- “NLMNLAEFD Format” on page 189

NLMNIAUD Format

Writesthe monetary format of the international expression for Australia.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

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.

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:
NLMNIBGN Format

Writes the monetary format of the international expression for Bulgaria.

**Category:** Numeric  
**Alignment:** Left

### Syntax

```plaintext
NLMNIBGN 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,nlmnibgn32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td><code>(BGN1,234.57)</code></td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td><code>$-1,234.57$</code></td>
</tr>
</tbody>
</table>

### See Also

Format:
- “NLMNLBGN Format” on page 191
**NLMNIBRL Format**

Writes the monetary format of the international expression for Brazil.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNIBRL}w.d \]

### 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 LOCALE= system option is set to English_UnitedStates.

```bash
x=put(-1234.56789,nlmnibrl32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BRL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLBRL Format” on page 192
NLMNICAD Format

Writes the monetary format of the international expression for Canada.

Category: Numeric
Alignment: Left

Syntax

NLMNICAD\(w.d\)

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.

\[x=\text{put}(-1234.56789,\text{nlmnicad32.2});\]
\[y=\text{put}(-1234.56789,\text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x=;)</td>
<td>(CAD1,234.57)</td>
</tr>
<tr>
<td>(y=;)</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLCAD Format” on page 193
**NLMNICHF Format**

Writes the monetary format of the international expression for Liechtenstein and Switzerland.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

```
NLMNICHFw.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,nlmnichf32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CHF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLCHF Format” on page 194
NLMNICNY Format

Writes the monetary format of the international expression for China.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNICNY\(w.d\)

**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: 02
- Range: 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmnicny32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>[CNY1,234.57]</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLNCY Format” on page 195
NLMNICZK Format

Writes the monetary format of the international expression for the Czech Republic.

- **Category:** 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.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmniczk32.2}); \\
y &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>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:**

- “NLMNLCZK Format” on page 196
NLMNIDKK Format

Writes the monetary format of the international expression for Denmark, Faroe Island, and Greenland.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNIDKK\(_w,d\)

**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,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:**

- “NLMNLDDKK Format” on page 197
**NLMNIEEK Format**  
Writes the monetary format of the international expression for Estonia.  

- **Category:** Numeric  
- **Alignment:** Left

---

**Syntax**

\[ \text{NLMNIEEK}w.d \]

**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`.  

\[
\begin{align*}
\text{x} &= \text{put}(-1234.56789, \text{nlmnieek}32.2) ; \\
\text{y} &= \text{put}(-1234.56789, \text{dollar32.2}) ;
\end{align*}
\]

<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**

- “NLMNIEEK Format” on page 198
NLMNIEGP Format

Writes the monetary format of the international expression for Egypt.

**Category:** Numeric

**Alignment:** Left

### Syntax

NLMNIEGP \(_{w,d}\)

### 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:** 3
- **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmniegp32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EGP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLEGP Format” on page 199
NLMNIEUR 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.

Category: Numeric
Alignment: Left

Syntax

NLMNIEUR\textsuperscript{w,d}

Syntax Description

\textit{w} \hspace{2cm} specifies the width of the output field.

Default \hspace{2cm} 12
Range \hspace{2cm} 8–32

\textit{d} \hspace{2cm} specifies the number of digits to the right of the decimal point in the numeric value.

Default \hspace{2cm} 2
Range \hspace{2cm} 0–28

Example

In the following example, the LOCALE= system option is set to \texttt{Locale=German\_Germany}.

\begin{verbatim}
x=put(-1234.56789,nlmnieur32.2);
y=put(-1234.56789,nlmnieur32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put(-1234.56789,nlmnieur32.2);</td>
<td>x=-1.234,57 EUR</td>
</tr>
<tr>
<td>y=put(-1234.56789,nlmnieur32.2);</td>
<td>y=-1.234,57 €</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLEUR Format” on page 200
NLMNIGBP Format

Writes the monetary format of the international expression for the United Kingdom.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\texttt{NLMNIGBPw.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 English\_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmnigbp32.2);
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>{GBP1,234.57}</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLMNLGBP Format” on page 201
**NLMNIHKD Format**

Wrote the monetary format of the international expression for Hong Kong.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

NLMNIHKD\textsubscript{\(w,d\)}

**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.  

\[x = \text{put}(-1234.56789, \text{nlnihkd32.2});\]
\[y = \text{put}(-1234.56789, \text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=);</td>
<td>HKD1,234.57</td>
</tr>
<tr>
<td>put (y=);</td>
<td>$1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “NLMNHLKD Format” on page 202
**NLMNIHRK Format**

Writes the monetary format of the international expression for Croatia.

- **Category:** Numeric
- **Alignment:** Left

---

**Syntax**

\[ \text{NLMNIHRK}_{w.d} \]

**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`.

\[ x=\text{put}(-1234.56789,\text{nlmnihrk32.2}); \]
\[ y=\text{put}(-1234.56789,\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x=; )</td>
<td>(HRK1,234.57)</td>
</tr>
<tr>
<td>put ( y=; )</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

- “NLMNLHRK Format” on page 203
**NLMNIHUF Format**

Writes the monetary format of the international expression for Hungary.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNIHUF} w.d \]

**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`.

\[
\begin{align*}
\text{x} &= \text{put}(-1234.56789, \text{nlmnihuf}32.2) ; \\
\text{y} &= \text{put}(-1234.56789, \text{dollar}32.2) ;
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put x=} ;</td>
<td>(HUF1,234.57)</td>
</tr>
<tr>
<td>\text{put y=} ;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “NLMNLHUF Format” on page 204
NLMNIIDR Format

Writes the monetary format of the international expression for Indonesia.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNIIDR\(w.d\)

**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.

\[ x = \text{put}(-1234.56789, \text{nlnstdr}32.2); \]
\[ y = \text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(IDR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

Format:

- “NLMNLIDR Format” on page 205
NLMNIILS Format

Writes the monetary format of the international expression for Israel.

Category: Numeric
Alignment: Left

Syntax

\[ \text{NLMNIILS} w.d \]

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.

\[ x = \text{put}(-1234.56789, \text{nlmniils}32.2); \]
\[ y = \text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ILS1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLILS Format” on page 206
NLMNIINR Format

Writes the monetary format of the international expression for India.

**Category:** Numeric
**Alignment:** Left

Syntax

NLMNIINR\(w,d\)

**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.

\(x=\text{put}(-1234.56789,\text{nlmniinr32.2});\)
\(y=\text{put}(-1234.56789,\text{dollar32.2});\)

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=;)</td>
<td>([\text{INR}1,234.57])</td>
</tr>
<tr>
<td>put (y=;)</td>
<td>(-1,234.57)</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**
- “NLMNLINR Format” on page 207
**NLMNIJPY Format**

Writes the monetary format of the international expression for Japan.

- **Category:** Numeric
- **Alignment:** Left

## Syntax

NLMNIJPY\(w,d\)

### 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_UnitedStates.

```plaintext
x=put(-1234.56789,nlmni32.2);
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:**

- “NLMNLJPY Format” on page 208
NLMNIKRW Format
 Writes the monetary format of the international expression for South Korea.

| Category: | Numeric |
| Alignment: | Left |

Syntax

NLMNIKRWw,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.

\[x=\text{put}(-1234.56789,\text{nlnmikrw32.2});\]
\[y=\text{put}(-1234.56789,\text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{put } x=;)</td>
<td>(\langle \text{KRW1,234.57}\rangle)</td>
</tr>
<tr>
<td>(\text{put } y=;)</td>
<td>($-1,234.57$)</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLKRW Format” on page 209
NLMNILTL Format

Writes the monetary format of the international expression for Lithuania.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNILTL\( w.d \)

**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.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmniltl32.2}); \\
y &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----+</td>
<td></td>
</tr>
<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:**

- “NLMNLLTL Format” on page 210
**NLMNILVL Format**

Writes the monetary format of the international expression for Latvia.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

**NLMNILVL**\(w.d\)

### 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.

```plaintext
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:**

- “NLMNLLVL Format” on page 211
NLMNIMOP Format

Writes the monetary format of the international expression for Macau.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

```
NLMNIMOPw,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,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**

- “NLMNLMOP Format” on page 212
NLMNIMXN Format

Writes the monetary format of the international expression for Mexico.

Category: 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=put(-1234.56789,nlmnimx32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MXN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLMXN Format” on page 213
NLMNIMYR Format

Writes the monetary format of the international expression for Malaysia.

**Category:** 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 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.

\[ \begin{align*} 
  x &= \text{put}(-1234.56789,\text{nlmnimyr}32.2) ; \\
  y &= \text{put}(-1234.56789,\text{dollar}32.2) ; \\
\end{align*} \]

<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:**

- “NLMNLMYR Format” on page 214
**NLMNINOK Format**

Writes the monetary format of the international expression for Norway.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNINOK \(w,d\)

**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.

\[x=\text{put}(-1234.56789,\text{nlmninok}\,32.2)\];
\[y=\text{put}(-1234.56789,\text{dollar}\,32.2)\];

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NOK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNINOK Format” on page 215
NLMNINZD Format

Writes the monetary format of the international expression for New Zealand.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNINZDw,d}

\textbf{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 the number of digits to the right of the decimal point in the numeric value.

\begin{itemize}
  \item Default: 2
  \item Range: 0–28
\end{itemize}

\textbf{Example}

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
  x=put(-1234.56789,nlmninzd32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{center}
\begin{tabular}{|c|c|}
  \hline
  \textbf{Statements} & \textbf{Results} \\
  \hline
  put x=; & (NZD\textdollar\textsterling1,234.57) \\
  put y=; & $-1,234.57$ \\
  \hline
\end{tabular}
\end{center}

\textbf{See Also}

Format:

\begin{itemize}
  \item “NLMNLNZD Format” on page 216
\end{itemize}
NLMNIIPLN Format

Writes the monetary format of the international expression for Poland.

Category: Numeric
Alignment: Left

Syntax

NLMNIIPLNw.d

Syntax Description

w

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>8–32</td>
</tr>
</tbody>
</table>

d

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

<table>
<thead>
<tr>
<th>Default</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>0–28</td>
</tr>
</tbody>
</table>

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:

• “NLMNLPLN Format” on page 217
NLMNIRUB Format

Writes the monetary format of the international expression for Russia.

**Category:** Numeric  
**Alignment:** Left

### Syntax

NLMNIRUB\(w.d\)

### 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.

\[x=\text{put}(-1234.56789,\text{nlmnirub32.2});\]
\[y=\text{put}(-1234.56789,\text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(RUB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**  
- “NLMNLRUB Format” on page 218
### NLMNISEK Format

Writes the monetary format of the international expression for Sweden.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

#### Syntax

\[ \text{NLMNISEK}_w.d \]

#### 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,nlmnisek32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SEK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

#### See Also

- “NLMNLSEK Format” on page 219
NLMNISGD Format

Writes the monetary format of the international expression for Singapore.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNISGD}\_w.d \]

**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,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:**

- “NLMNLSGD Format” on page 220
NLMNITHB Format

Writes the monetary format of the international expression for Thailand.

Category: Numeric
Alignment: Left

Syntax

NLMNITHBw.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, nlmnithb32.2);
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:

• “NLMNLTTHB Format” on page 221
**NLMNITRY Format**

Writes the monetary format of the international expression for Turkey.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNITRY}^{w,d} \]

**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.

\[
\begin{align*}
\text{x=put(-1234.56789,nlmnity32.2);} \\
\text{y=put(-1234.56789,dollar32.2);} \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TRY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “**NLMNLTRY Format**” on page 222
**NLMNITWD Format**

Writes the monetary format of the international expression for Taiwan.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNITWD}_w,d \]

**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.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnitwd32.2}); \\
y &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TWD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$,-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNLTWD Format” on page 223
**NLMNIUSD Format**

 Writes the monetary format of the international expression for Puerto Rico and the United States.

 **Category:** Numeric  
 **Alignment:** Left

---

**Syntax**

NLMNIUSD\(_w,d\)

**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,nlmniusd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(USD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLUSD Format” on page 224
NLMNIZAR Format

Writes the monetary format of the international expression for South Africa.

**Category:** Numeric
**Alignment:** Left

### Syntax

\[ \text{NLMNIZAR}^{w.d} \]

### 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,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>(\text{ZAR}1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “\texttt{NLMNLZAR Format}” on page 225
**NLMNLAED Format**

Writes the monetary format of the local expression for the United Arab Emirates.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNLAED} w.d \]

**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:** 3
  - **Range:** 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
  x & = \text{put}(-1234.56789, \text{nlmnlaed}32.2); \\
  y & = \text{put}(-1234.56789, \text{dollar}32.2);
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AED(1), 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “NLMNIAED Format” on page 152
NLMNLAUD Format

Writes the monetary format of the local expression for Australia.

Category: Numeric
Alignment: Left

Syntax

NLMNLAUDw.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,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 Format” on page 153
NLMNLBGN Format

Writes the monetary format of the local expression for Bulgaria.

Category: Numeric
Alignment: Left

Syntax

NLMNLBGN\_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=put(-1234.56789,nlmnlbgn32.2);
\_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:

• “NLMNIBGN Format” on page 154
NLMNLBRL Format

Writes the monetary format of the local expression for Brazil.

Category: 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.

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:
- “NLMNIBRL Format” on page 155
NLMNLCAD Format

Writes the monetary format of the local expression for Canada.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLCAD\(w.d\)

**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.

\[x=\text{put}(-1234.56789,\text{nlmnlcad32.2});\]
\[y=\text{put}(-1234.56789,\text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+-------+</td>
<td></td>
</tr>
<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:**

- “NLMNICAD Format” on page 156
NLMNLCHF Format

Writes the monetary format of the local expression for Liechtenstein and Switzerland.

Category: Numeric
Alignment: Left

Syntax

NLMNLCHFw.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.

\[x=\text{put}(-1234.56789,\text{nlmnlchf32.2});\]
\[y=\text{put}(-1234.56789,\text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>SFr.1,234.57</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNICHF Format” on page 157
NLMNLCNY Format

Writes the monetary format of the local expression for China.

Category: Numeric
Alignment: Left

Syntax

NLMNLCNYw.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,nlmnlcny32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>[RMB1,234.57]</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNICNY Format” on page 158
NLMNLCZK Format

Writes the monetary format of the local expression for the Czech Republic.

**Category:** Numeric  
**Alignment:** Left

## Syntax

\[
\text{NLMNLCZK} \ w.d
\]

### 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.

```plaintext
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:**  
- “NLMNICZK Format” on page 159
NLMNLDKK Format

Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLDKKw.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}.

\texttt{x=put(-1234.56789,nlmnlk32.2);}
\texttt{y=put(-1234.56789,dollar32.2);}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{x=};</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>\texttt{y=};</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIDKK Format” on page 160
NLMNLEEK Format

Writes the monetary format of the local expression for Estonia.

Category: Numeric
Alignment: Left

Syntax

NLMNLEEK\textit{w.d}

Syntax Description

\textit{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.

\begin{verbatim}
x=put(-1234.56789,nlmnleek32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{put~} ) ( x=; )</td>
<td>( (Kr1,234.57) )</td>
</tr>
<tr>
<td>( \text{put~} ) ( y=; )</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIEEK Format” on page 161
NLMNLEGP Format

Writes the monetary format of the local expression for Egypt.

Category: Numeric
Alignment: Left

Syntax

NLMNLEGP<sub>w,d</sub>

Syntax Description

<sub>w</sub>

specifies the width of the output field.
Default: 12
Range: 8–32

<sub>d</sub>

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.

```plaintext
x=put(-1234.56789,nlmnlegp32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EGP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIEGP Format” on page 162
**NLMNLEUR 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.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLEUR}^{w.d} \]

**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 German_Germany.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnieur32}.2) ; \\
y &= \text{put}(-1234.56789, \text{nlmnieur32}.2) ;
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td>( x=-1.234,57 \text{ EUR} )</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>( y=-1.234,57 \text{ €} )</td>
</tr>
</tbody>
</table>

---

**See Also**

- “NLMNIEUR Format” on page 163
NLMNLGBP Format

Writes the monetary format of the local expression for the United Kingdom.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNLGBP}w.d \]

### 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**.

\[ x=\text{put}(-1234.56789,\text{nlmnlgbp}32.2); \]
\[ y=\text{put}(-1234.56789,\text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \textit{x} =;</td>
<td>(£1,234.57)</td>
</tr>
<tr>
<td>put \textit{y} =;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNIGBP Format” on page 164
NLMNLHKD Format

Writes the monetary format of the local expression for Hong Kong.

**Category:** Numeric

**Alignment:** Left

### Syntax

\[
\text{NLMNLHKD}w.d
\]

### 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`.

\[
x = \text{put}(-1234.56789, \text{nlmnlhkd32.2});
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HK$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNIHKD Format” on page 165
**NLMNLHRK Format**

Writes the monetary format of the local expression for Croatia.

- **Category:** Numeric
- **Alignment:** Left

---

**Syntax**

\[ \text{NLMNLHRK}w.d \]

**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,nlmnlhrk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Kn1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

- [“NLMNIHRK Format” on page 166](#)
NLMNLHUF Format

Writes the monetary format of the local expression for Hungary.

**Category:** Numeric

**Alignment:** Left

---

**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 LOCALE= system option is set to English_UnitedStates.

\[x = \text{put}(-1234.56789, \text{nlmnlhuf32.2});\]
\[y = \text{put}(-1234.56789, \text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=);</td>
<td>((\text{Ft}1,234.57))</td>
</tr>
<tr>
<td>put (y=);</td>
<td>$\text{-}1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNIHUF Format” on page 167
**NLMNLIDR Format**

Writes the monetary format of the local expression for Indonesia.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNLIDR} w.d \]

**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.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnlidr32.2}) \\
y &= \text{put}(-1234.56789, \text{dollar32.2})
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put} x=;</td>
<td>(Rp1,234.57)</td>
</tr>
<tr>
<td>\text{put} y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNIIDR Format” on page 168
NLMNLILS Format

Writes the monetary format of the local expression for Israel.

Category: Numeric  
Alignment: Left

Syntax

NLMNLILSw.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. 

\( x=\text{put}(-1234.56789,\text{nlmnlil}32.2); \)
\( y=\text{put}(-1234.56789,\text{dollar}32.2); \)

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>[ILS1,234.57]</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIILS Format” on page 169
NLMNLINR Format

Writes the monetary format of the local expression for India.

<table>
<thead>
<tr>
<th>Category</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLMNLINR\(w.d\)

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.

\[x=\text{put}(-1234.56789,\text{nlmnlinr}32.2)\];
\[y=\text{put}(-1234.56789,\text{dollar}32.2)\];

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------+---------------------</td>
</tr>
<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:

- “NLMNIIINR Format” on page 170
**NLMNLJPY Format**

Writes the monetary format of the international expression for Japan.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNLJPY}^w.d \]

### 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_UnitedStates.

\[ x = \text{put}(\text{-1234.56789}, \text{nlmnljpy32.2}); \]
\[ y = \text{put}(\text{-1234.56789}, \text{dollar32.2}); \]

<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:**
- “NLMNIJPY Format” on page 171
NLMNLRW Format

NLMNLRW writes the monetary format of the local expression for South Korea.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLRW<sub>w,d</sub>

**Syntax Description**

- **<sub>w</sub>**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **<sub>d</sub>**
  - 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.

```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:**

- "NLMNIKRW Format" on page 172
**NLMNLLTL Format**

Writes the monetary format of the local expression for Lithuania.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLLTL}\ w.d \]

**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_.

```plaintext
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:**

- “NLMNILTL Format” on page 173
NLMNLLVL Format

Writes the monetary format of the local expression for Latvia.

Category: Numeric
Alignment: Left

Syntax

NLMNLLVL<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=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:

• “NLMNIVL Format” on page 174
NLMNLMO\textsubscript{P} Format

Writes the monetary format of the local expression for Macau.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\texttt{NLMNLMO}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 English\_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmnlmop32.2);
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>$(P1,234.57)$</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

### See Also

- “NLMNIMOP Format” on page 175
NLNMNX Format

Writes the monetary format of the local expression for Mexico.

**Category:** Numeric

**Alignment:** Left

## Syntax

NLNMNX\(w.d\)

### Syntax Description

\(w\)

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>

\(d\)

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>2</td>
<td>0–28</td>
</tr>
</tbody>
</table>

## Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmlmxn32.2);
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:**

- “NLNMIMXN Format” on page 176
NLMNLMYR Format

Writes the monetary format of the local expression for Malaysia.

Category: Numeric
Alignment: Left

Syntax

NLMNLMYR_{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.

\[ x = \text{put}(-1234.56789, \text{nlmnlmyr32.2}); \]
\[ y = \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x=; )</td>
<td>(R1,234.57)</td>
</tr>
<tr>
<td>put ( y=; )</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIMYR Format” on page 177
NLMNLNOK Format

Writes the monetary format of the local expression for Norway.

**Category:** Numeric  
**Alignment:** Left

## Syntax

\[ \text{NLMNLNOK}\_w.d \]

### 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.

\[
x = \text{put}(-1234.56789, \text{nlmnlnok32.2});
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put } x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>\text{put } y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLMNINOK Format” on page 178
**NLMNLNZD Format**

Writes the monetary format of the local expression for New Zealand.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[\text{NLMNLNZD}w.d\]

**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 `LOCASE=` system option is set to `English_UnitedStates`.

\[
x=\text{put}(-1234.56789,\text{nlmnlnzd32.2});
y=\text{put}(-1234.56789,\text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=;)</td>
<td>(NZ$1,234.57)</td>
</tr>
<tr>
<td>put (y=;)</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNINZD Format” on page 179
NLMNLPLN Format

Writes the monetary format of the local expression for Poland.

Category: Numeric
Alignment: Left

Syntax

NLMNLPLNw.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.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnlpln32.2}) \\
y &= \text{put}(-1234.56789, \text{dollar32.2})
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(PLN, 234.57</td>
</tr>
<tr>
<td>put x=;</td>
<td>$-1,234.57</td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIPLN Format” on page 180
NLMNLRUB Format

Writes the monetary format of the local expression for Russia.

Category: Numeric
Alignment: Left

Syntax

NLMNLRUBw.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,nlmnlrub32.2);
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:

• “NLMNIRUB Format” on page 181
NLMNLSEK Format

Writes the monetary format of the local expression for Sweden.

**Category:** Numeric  
**Alignment:** Left

### Syntax

NLMNLSEK\(w,d\)

### 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,nlmnlsek32.2);
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:**

- “NLMNISEK Format” on page 182
NLMNLSGD Format

Writes the monetary format of the local expression for Singapore.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

NLMNLSGD\(w,d\)

**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.

\(x = \text{put}(-1234.56789, \text{nlmnlsgd}32.2);\)
\(y = \text{put}(-1234.56789, \text{dollar}32.2);\)

<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**

- “NLMNISGD Format” on page 183
NLMNLTHB Format

Writes the monetary format of the local expression for Thailand.

Category: Numeric
Alignment: Left

Syntax

\[
\text{NLMNLTHB} \ w.d
\]

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_UnitedStates.

\[
x=\text{put}(-1234.56789, \text{nlmnltb}32.2);
y=\text{put}(-1234.56789, \text{dollar}32.2);
\]

<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:

- “NLMNITHB Format” on page 184
**NLMNLTRY Format**

Writes the monetary format of the local expression for Turkey.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLTRY} w.d \]

**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`.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnltry}32.2); \\
y &= \text{put}(-1234.56789, \text{dollar}32.2);
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(YTL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNITRY Format” on page 185
NLMNLTWD Format

 Writes the monetary format of the local expression for Taiwan.

 Category: Numeric
 Alignment: Left

 Syntax

 NLMNLTWDw.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,nlmnltd32.2);
y=put(-1234.56789,dollar32.2);

 Statements Results

----+--------+-
put x=; (NT$1,234.57)
put y=; $-1,234.57

 See Also

 Format:
 • “NLMNITWD Format” on page 186
NLMNLUSD Format

Writes the monetary format of the local expression for Puerto Rico and the United States.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLUSD}_{w.d} \]

**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.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnlusd32.2}); \\
y &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>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:**

- “NLMNIUSD Format” on page 187
NLMNLZAR Format

Writes the monetary format of the local expression for South Africa.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[ \text{NLMNLZAR} \ w, d \]

### 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.

\[ \begin{align*} 
  x &= \text{put}(-1234.56789, \text{nlnmlzar32.2}); \\
  y &= \text{put}(-1234.56789, \text{dollar32.2}); 
\end{align*} \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>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:**

- “NLMNIZAR Format” on page 188
NLMNY Format

Writes the monetary format of the local expression in the specified locale using local currency.

**Categories:** Numeric

**Alignment:** Left

**Syntax**

\[ \text{NLMNY}w.d \]

**Syntax Description**

\( w \)

- 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 NLMNY\( w.d \) informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMNY\( w.d \) format writes numeric values by using the currency symbol, the thousands separator, and the decimal separator that is used by the locale.

*Note:* The NLMNY\( w.d \) format does not convert currency format. Therefore, the value of the formatted number should equal the currency of the current locale value.

**Comparisons**

The NLMNY\( w.d \) and NLMNYI\( w.d \) formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYI\( w.d \) format uses three-letter international currency codes, such as USD, while NLMNY\( w.d \) format uses local currency symbols, such as $.

The NLMNY\( w.d \) format is similar to the DOLLAR\( w.d \) format except that the NLMNY\( w.d \) format is locale-specific.

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmny32.2);
y=put(-1234.56789, dollar32.2);
```
Statements | Results
---+----1----+
put x=; | $(1,234.57)
put y=; | $-1,234.57

See Also

Format:
- “NLMNYI Format” on page 227

Informs:
- “NLMNY Informat” on page 529
- “NLMNYI Informat” on page 530

NLMNYI Format

 Writes the monetary format of the international expression in the specified locale.

**Categories:** Numeric, CAS

**Alignment:** Left

**Syntax**

NLMNYIw.d

**Syntax Description**

w

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.

\[
x=\text{put}(-1234.56789,\text{nlmnyi32.2});
y=\text{put}(-1234.56789,\text{nlmny32.2});
z=\text{put}(-1234.56789,\text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=);</td>
<td>(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:

- “NLMNY Format” on page 226

Informats:

- “NLMNY Informat” on page 529
  - “NLMNYI Informat” on page 530

NLNUM Format

Writes the numeric format of the local expression in the specified locale.

Categories: Numeric  CAS

Alignment: Left

Syntax

\[\text{NLNUM}w.d\]
**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 NLMUM\( w,d \) informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMUM\( w,d \) format writes numeric values by using the thousands separator and the decimal separator that is used by the locale.

**Comparisons**

The NLMUM\( w,d \) format writes the numeric value with locale-dependent thousand and decimal separators. The NLMUMI\( w,d \) 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 NLMUM\( w,d \) format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

**Example**

\[ x=put(-1234356.7891,nlnum32.2); \]

<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:**
- “NLNUMI Format” on page 230
Informats:
• “NLNUM Informat” on page 531
• “NLNUMI Informat” on page 533

NLNUMI Format

Writes the numeric format of the international expression in the specified locale.

Categories: Numeric
            CAS
Alignment: Left

Syntax

NLNUMI_{w.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
Range 0–31

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

\[x = \text{put}(-1234356.7891, \text{nlnumi32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{options LOCALE=English_UnitedStates;}</td>
<td>\text{-1,234,356.79}</td>
</tr>
<tr>
<td>\text{put } x=;</td>
<td>\text{-1,234,356.79}</td>
</tr>
<tr>
<td>\text{options LOCALE=German_Germany;}</td>
<td>\text{-1,234,356.79}</td>
</tr>
<tr>
<td>\text{put } x=;</td>
<td>\text{-1,234,356.79}</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLNUM Format” on page 228

Informats:

- “NLNUM Informat” on page 531
- “NLNUMI Informat” on page 533

**NLPCT Format**

Writes percentage data of the local expression in the specified locale.

**Categories:**

- Numeric
- CAS

**Alignment:**

- Left

**Syntax**

\[\text{NLPCTw}.d\]

**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.
The NLPCTI\.d format writes percentage data of the international expression in the specified locale. The NLPCT\.d format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTI\.d format writes the percentage value with a comma (,) as thousands separator and a period (.) as a decimal separator.

The NLPCT\.d format is similar to the PERCENT\.d format except the NLPCT\.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>option 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>
<tr>
<td>option 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:**
- “NLPCTI Format” on page 232

**Informats:**
- “NLPCT Informat” on page 534
- “NLPCTI Informat” on page 535

**NLPCTI Format**

Writes percentage data of the international expression in the specified locale.

**Categories:** Numeric
Syntax

**NLPCTI**\(w.d\)

**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 NLPCTI\(w.d\) format writes percentage data of the international expression in the specified locale. The NLPCT\(w.d\) format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTI\(w.d\) format writes the percentage value with a comma (,) as thousands separator and a period (.) as a decimal separator.

The NLPCT\(w.d\) format is similar to the PERCENT\(w.d\) format except the NLPCT\(w.d\) format is locale-specific.

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{put}(-12.3456789, \text{nlpcti}32.2); \\
y &= \text{put}(-12.3456789, \text{percent}32.2);
\end{align*}
\]

<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:
NLPCTN Format

Produces percentages, using a minus sign for negative values.

Categories:
- Numeric
- CAS

Alignment: Right

Syntax

NLPCTN\textsubscript{w,d}

Syntax Description

\( w \)

specifies the width of the output field.

- Default: 6
- Range: \( 4 \leq w \leq 32 \)
- 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.

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value. This argument is optional.

- Default: 0
- Range: \( 0 \leq d \leq 31 \)
- Requirement: must be less than \( w \)

Details

The NLPCTN\textsubscript{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>( x = -0.02; )</td>
<td></td>
</tr>
</tbody>
</table>
NLPCTP Format

Writes locale-specific numeric values as percentages.

**Categories:** Numeric

**CAS**

**Alignment:** Right

**Syntax**

NLPCTP<sub>w,d</sub>

**Syntax Description**

<sub>w</sub>  
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 (%).

<sub>d</sub>  
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 <sub>w</sub>

**Details**

The NLPCTP<sub>w,d</sub> format multiplies values by 100, formats them, and adds a percent sign (%) to the end of the formatted value. The NLPCTP<sub>w,d</sub> format is similar to the PERCENT<sub>w,d</sub> format except that the thousands separator and decimal symbol for the NLPCTP<sub>w,d</sub> format is locale-specific.

**Example**

```plaintext
x=-0.02;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=-2%</td>
<td></td>
</tr>
</tbody>
</table>
NLPVALUE Format

Writes p-values of the local expression in the specified locale.

Categories: Numeric, CAS

Alignment: Left

Syntax

NLPVALUE\(w,d\)

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:

```plaintext
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 pvalue1.1 +5 x nlpvalue1.1;  
  put x= +5 x pvalue20.2 +5 x nlpvalue20.2;  
  put x= +5 x pvalue32.3 +5 x nlpvalue32.3;  
run;  
```
See Also

Format:
• “PVALUE Format” in SAS Viya Formats and Informats: Reference

NLSTRMON Format

Writes the month name in the specified locale.

Categories: Numeric
CAS

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 eighth month (August)
- 9 = the ninth month (September)
- 10 = the tenth month (October)
- 11 = the eleventh month (November)
- 12 = the twelfth 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>January</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.;</td>
<td>Jan</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.1; /* decimal .1 specified use abbreviation. */</td>
<td>JANUARY</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.2;</td>
<td>JAN</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.3;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLSTRQTR Format**

Writes a numeric value as the quarter-of-the-year in the specified locale.

**Categories:** Numeric

**CAS:**

**Alignment:** Left

**Syntax**

\texttt{NLSTRQTRw.d}

**Syntax Description**

\texttt{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 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; run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLSTRWK Format**

Writes a numeric value as the day-of-the-week in the specified locale.

**Categories:**
- Numeric
- CAS

**Alignment:**
- Left
Syntax
NLSTRWK\textit{w}.\textit{d}

Syntax Description
\textit{w}

specifies the width of the output field

Default 20

Range 1–200

\textit{d}

specifies the following:

- 00000001: write abbreviated form.
- 00000010: write capitalized form.

Default 0

Range 0–3

Details

The NLSTRWK\textit{w}.\textit{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>;</td>
<td>Sunday</td>
</tr>
<tr>
<td>wknum = 1; /* Sunday=1, Saturday=7 */</td>
<td>Sun</td>
</tr>
<tr>
<td>put wknum NLSTRWK20. ;</td>
<td>SUNDAY</td>
</tr>
<tr>
<td>put wknum NLSTRWK20.1; /* decimal .1</td>
<td>SUN</td>
</tr>
<tr>
<td>specified use abbreviation. */</td>
<td></td>
</tr>
<tr>
<td>put wknum NLSTRWK20.2;</td>
<td></td>
</tr>
<tr>
<td>put wknum NLSTRWK20.3;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
NLTIMAP 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.

Categories: Date and Time  
          CAS
Alignment: Left

Syntax

NLTIMAP{w}.

Syntax Description

w
  specifies the width of the output field.
  Default 10
  Range 4–200

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 60 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>options locale=German_Germany;</td>
<td>4:24:43 nachm</td>
</tr>
<tr>
<td>put time nltimap14.;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Format:
- “NLTIME Format” on page 242

**NLTIME 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:** Date and Time

**CAS**

**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.

<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 nltim.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>16.24</td>
</tr>
<tr>
<td>put time nltim.;</td>
<td></td>
</tr>
</tbody>
</table>
$UCS2B 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

### Syntax

\$UCS2Bw:

### Syntax Description

\( w \)

\( w \) 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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1</td>
<td></td>
</tr>
</tbody>
</table>
data _null_; 
  x = 'Å'; 
  y = put(x,$ucs2b2.); 
  put y $hex.; 
run;

See Also

Formats:
- “$UCS2L Format” on page 245
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 264
- “$UCS2BE Format” on page 244

Informats:
- “$UCS2B Informat” on page 538
- “$UCS2BE Informat” on page 539
- “$UCS2L Informat” on page 540
- “$UCS2X Informat” on page 543
- “$UTF8X Informat” on page 557

$UCS2BE 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

Syntax

$UCS2BEw.

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 $UCS2BEw. 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 $UCS2BEw. format performs processing that is the opposite of the $UCS2Bw. format.

Example
This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '592700410042'x; put x $ucs2be.;</td>
<td>AB</td>
</tr>
</tbody>
</table>

See Also
Formats:
- “$UCS2B Format” on page 243

Informats:
- “$UCS2B Informat” on page 538
- “$UCS2BE Informat” on page 539

$UCS2L 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.

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.
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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>2759</td>
</tr>
<tr>
<td>x = 'ä';</td>
<td></td>
</tr>
<tr>
<td>y=put(x,$ucs2l2.);</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:
- “$UCS2B Format” on page 243
- “$UCS2LE Format” on page 247
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 264

Informats:
- “$UCS2B Informat” on page 538
- “$UCS2L Informat” on page 540
- “$UCS2LE Informat” on page 542
- “$UCS2X Informat” on page 543
- “$UTF8X Informat” on page 557
$UCS2LE 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

### 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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = '275941004200'x;</code></td>
<td>AB</td>
</tr>
<tr>
<td>put x $ucs2le.;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Format:**  
- “$UCS2L Format” on page 245
Informats:
- “$UCS2L Informat” on page 540
- “$UCS2LE Informat” on page 542

$UCS2X 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

Syntax
$UCS2Xw.

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 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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '大';</td>
<td>'5927'x (binary) or '2759'x (little endian)</td>
</tr>
<tr>
<td>put x $ucs2x2.;</td>
<td></td>
</tr>
</tbody>
</table>
$UCS2XE 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

### Syntax

$UCS2XEw.

### 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 $UCS2XE Item. 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 $UCS2XE Item. format performs processing that is the opposite of the $UCS2X Item. format.
Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 'e5a4a7'x;</td>
<td></td>
</tr>
<tr>
<td>put x $utf8xe10.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- “$UCS2X Format” on page 248

Informats:
- “$UCS2X Informat” on page 543
- “$UCS2XE Informat” on page 544

$UCS4B 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

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.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>4–32767</td>
</tr>
</tbody>
</table>

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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = '¥';$</td>
<td>'00005927'x (binary)</td>
</tr>
<tr>
<td>put $x$ $ucs4b4.;$</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- “$UCS2L Format” on page 245
- “$UCS2X Format” on page 248
- “$UCS4BE Format” on page 251
- “$UCS4L Format” on page 252
- “$UCS4X Format” on page 255
- “$UTF8X Format” on page 264

Informats:

- “$UCS2B Informat” on page 538
- “$UCS2L Informat” on page 540
- “$UCS2X Informat” on page 543
- “$UCS4B Informat” on page 545
- “$UCS4L Informat” on page 546
- “$UCS4X Informat” on page 547
- “$UTF8X Informat” on page 557

$UCS4BE 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
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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = ’000059270000004100000042’x; put x $ucs4be.;</td>
<td>⬇AB</td>
</tr>
</tbody>
</table>

See Also

Format:

• “$UCS4B Format” on page 250

Informat:

• “$UCS4B Informat” on page 545

$UCS4L 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
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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>2759</td>
</tr>
<tr>
<td>x = '𠆝';</td>
<td></td>
</tr>
<tr>
<td>y=put(x,$ucs414.);</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:

- “$UCS2B Format” on page 243
- “$UCS2X Format” on page 248
- “$UCS4B Format” on page 250
- “$UCS4LE Format” on page 254
- “$UCS4X Format” on page 255
$UCS4LE 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

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 UTF-8 encoding.
Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UCS4X$</td>
<td>$AB$</td>
</tr>
</tbody>
</table>

```plaintext
x = '275900004100000042000000'x;
put x $ucs4le.;
```

See Also

Format:
- “$UCS4L Format” on page 252

Informat:
- “$UCS4L Informat” on page 546

$UCS4X 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
**Alignment:** Left

**Syntax**

$UCS4X$w:

**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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n x = '芊';</td>
<td>'00005927'x (binary) or '27590000'x (little endian)</td>
</tr>
</tbody>
</table>

put x $ucs4x4.;

See Also

Formats:
- “$UCS2L Format” on page 245
- “$UCS4XE Format” on page 256
- “$UCS2X Format” on page 248
- “$UCS4B Format” on page 250
- “$UCS4L Format” on page 252
- “$UTF8X Format” on page 264

Informats:
- “$UCS2B Informat” on page 538
- “$UCS2L Informat” on page 540
- “$UCS2X Informat” on page 543
- “$UCS4B Informat” on page 545
- “$UCS4B Format” on page 250
- “$UCS4L Informat” on page 546
- “$UCS4X Informat” on page 547
- “$UTF8X Informat” on page 557

$UCS4XE 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

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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '275900004100000042000000'x; put x $ucs4be4.;</td>
<td>AB (little endian)</td>
</tr>
</tbody>
</table>

See Also

Format:

- “$UCS4X Format” on page 255

Informat:

- “$UCS4X Informat” on page 547
Syntax
$UESC_w$

Syntax Description

\( w \)

specifies the width of the input 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. $UESC_w$ can be nested.

Comparisons

The $UESC_w$ format performs processing that is opposite of the $UESCE_w$ format.

Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='¥u5927' ;</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:

• “$UESCE Format” on page 259

Informats:

• “$UESC Informat” on page 549
• “$UESCE Informat” on page 550
$UESCE 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

### Syntax

$UESCE w$

### 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 $UESCE w$ format performs processing that is the opposite of the $UESC w$ format.

### Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put('¥u5927',$uesce10.) ;</td>
<td>x=Yu5927</td>
</tr>
<tr>
<td>x=put('¥uu5927',$uesce10.) ;</td>
<td>x=Yu5uu7</td>
</tr>
<tr>
<td>x=put('¥uuu5927',$uesce10.) ;</td>
<td>x=Yuuu5927</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “$UESC Format” on page 257

**Informats:**
$UNCR 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 |

**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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='91E5'x ; /* Japanese 'ér' in Shift-JIS */</td>
<td>&amp;22823</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:**

- “$UNCRE Format” on page 261
$UNCRE 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

**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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='&amp;22823;abc';</td>
<td>abc</td>
</tr>
<tr>
<td>put x $uncr10.;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

- “$UNCR Format” on page 260
Informs:
- “$UNCR Informat” on page 551
- “$UNCRE Informat” on page 552

$UPAREN Format
Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation.

Category: Character
Alignment: Left

Syntax
$UPARENw.

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 $UPARENw. format performs processing that is the opposite of the $UPARENEw. format.

Example
This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=' motorists';</td>
<td>&lt;u5927&gt;</td>
</tr>
<tr>
<td>y='abc3';</td>
<td>&lt;u0061&gt; &lt;u0062&gt; &lt;u0063&gt; &lt;u0033&gt;</td>
</tr>
<tr>
<td>put x $uparen7.;</td>
<td></td>
</tr>
<tr>
<td>put y $uparen28.;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:
- “$UPAREN Format” on page 263

Informs:
- “$UPAREN Informat” on page 553
- “$UPAREN Informat” on page 555

$UPAREN 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

Syntax
$UPARENw

Syntax Description
w
specifies the width of the output field.

Default 8
Range 1–32767

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

Example
This example uses UTF-8 encoding.

<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 $uparen4.;</td>
<td>abc3</td>
</tr>
</tbody>
</table>
$UTF8X 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

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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '91E5'x; /* Japanese '.AllArgsConstructor in Shift-JIS */</td>
<td>x='e5a4a7'x</td>
</tr>
<tr>
<td>put x $utf8x10.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

• “$UCS2B Format” on page 243
$UTF8XE 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

### Syntax

$UTF8XE_w$

### 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.

<table>
<thead>
<tr>
<th>Default</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range</strong></td>
<td>1–32767</td>
</tr>
</tbody>
</table>

### Comparisons

The $UTF8XE_w$ format performs processing that is the opposite to the $UTF8X_w$ format.

### Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = unicode('u5927');</td>
<td></td>
</tr>
<tr>
<td>put x $utf8xe10.;</td>
<td></td>
</tr>
</tbody>
</table>
WEEKU Format

Writes a week number in decimal format by using the U algorithm.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

\[ \text{WEEKU}_w. \]

**Syntax Description**

\[ w \]

specifies the width of the output field.

**Default** 11

**Range** 2–200

**Details**

The `WEEKU_w.` format writes a week-number format. The `WEEKU_w.` 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>yyyyWWW</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yyyyWWWDD</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWWWDD</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-WWdd</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:

- “WEEKV Format” on page 267
- “WEEKW Format” on page 269

WEEKV Format

WEEKVw writes a week number in decimal format by using the V algorithm.

Category: Date and Time
Alignment: Left

Syntax

WEEKVw:
**Syntax Description**

\[ w \]

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>2–200</td>
</tr>
</tbody>
</table>

**Details**

The WEEKV\(w\). 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>
<tr>
<td>7-8</td>
<td>yyWwwd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWwwd</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 WEEK\(Vw\). 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 WEEK\(Ww\). 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 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**

```sas
sasdate='01JAN2003'd;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---+-----1---+</td>
</tr>
</tbody>
</table>
WEEKW Format

WEEKW format writes a week number in decimal format by using the W algorithm.

**Syntax**

WEEKWw.

**Syntax Description**

w

specifies the width of the output field.

| Default | 11 |
| Range   | 2–200 |

**Details**

The WEEKWw. 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>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,weekv1.);</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

Formats:

- “WEEKU Format” on page 266
- “WEEKW Format” on page 269
Comparisons

The WEEKV<sub>w</sub> 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 WEEKW<sub>w</sub> 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 WEEKU<sub>w</sub> 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,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:
- “WEEKU Format” on page 266
- “WEEKV Format” on page 267
YEN Format

Writes numeric values with yen signs, commas, and decimal points.

- **Categories:** Numeric
  - CAS
- **Alignment:** Right

---

**Syntax**

\[
\text{YEN}w.d
\]

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - **Default:** 8
  - **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 \text{YEN}w.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>
YYWEEKU Format

Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.

Categories: Date and Time
CAS
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

sasdate = '01JAN2007'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweeku3.);</td>
<td>W00</td>
</tr>
<tr>
<td>v=put(sasdate,yyweeku4.);</td>
<td>W00</td>
</tr>
<tr>
<td>w=put(sasdate,yyweeku5.);</td>
<td>07W00</td>
</tr>
<tr>
<td>x=put(sasdate,yyweeku6.);</td>
<td>07W00</td>
</tr>
<tr>
<td>y=put(sasdate,yyweeku7.);</td>
<td>2007W00</td>
</tr>
<tr>
<td>z=put(sasdate,yyweeku8.);</td>
<td>2007-W00</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:

- “WEEKU Format” on page 266

YYWEEKV Format

Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.

Categories: Date and Time
CAS

Alignment: Left

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 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>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**

```sas
sasdate = '01JAN2007'd;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>w=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; put v; put w; put x; put y; put z;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Format:
- “WEEKV Format” on page 267
YY WEEKW Format

Writes a week number in decimal format by using the W algorithm, excluding the day-of-week information.

**Categories:** Date and Time
CAS

**Alignment:** Left

---

**Syntax**

YY WEEKWw.

**Syntax Description**

W

specifies the width of the output field.

Default 7

Range 2–8

---

**Details**

The YY WEEKWw. 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 YY WEEKWw. format is similar to the WEEKWw. format except that the YY WEEKWw. format does not specify the day-of-week information. Also, the YY WEEKWw. 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><code>u=put(sasdate,yyweekw3.);</code></td>
<td>W01</td>
</tr>
<tr>
<td><code>v=put(sasdate,yyweekw4.);</code></td>
<td>W01</td>
</tr>
<tr>
<td><code>w=put(sasdate,yyweekw5.);</code></td>
<td>07W01</td>
</tr>
<tr>
<td><code>x=put(sasdate,yyweekw6.);</code></td>
<td>07W01</td>
</tr>
<tr>
<td><code>y=put(sasdate,yyweekw7.);</code></td>
<td>2007W01</td>
</tr>
<tr>
<td><code>z=put(sasdate,yyweekw8.);</code></td>
<td>2007-W01</td>
</tr>
<tr>
<td><code>put u;</code></td>
<td></td>
</tr>
<tr>
<td><code>put v;</code></td>
<td></td>
</tr>
<tr>
<td><code>put w;</code></td>
<td></td>
</tr>
<tr>
<td><code>put x;</code></td>
<td></td>
</tr>
<tr>
<td><code>put y;</code></td>
<td></td>
</tr>
<tr>
<td><code>put z;</code></td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “WEEKW Format” on page 269
Part 5

Functions for NLS

Chapter 10
Internationalization Compatibility for SAS String Functions ........ 279

Chapter 11
VARCHAR Support in Functions ........................................ 299

Chapter 12
Dictionary of NLS Call Routines and Functions ................. 303
Chapter 10
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 string. The Table 10.1 on page 280 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. For SAS Viya, you should consider using the K functions to process your UTF-8 data.

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 10.1 on page 280

<table>
<thead>
<tr>
<th>Table 10.1</th>
<th>String Representation in UTF-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexadecimal Representation</td>
<td>E282AC</td>
</tr>
<tr>
<td></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>
<thead>
<tr>
<th>Table 10.2</th>
<th>Code Example Using Regular Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements</td>
<td>Results</td>
</tr>
<tr>
<td>/* SAS program to submit in a UTF-8 SAS session */</td>
<td>str=E282AC313233</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>s=sl=1 s=E22020202020</td>
</tr>
<tr>
<td>str= &quot;€123&quot; ;</td>
<td>l=6</td>
</tr>
<tr>
<td>s=substr(str,1,1) ;</td>
<td></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 10.3 on page 281 solves this issue by substituting the SUBSTR() function, which only works 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.
**Table 10.3** Code Example Using K 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 */</td>
<td></td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>str= &quot;€123&quot; ;</td>
<td>str=E282AC313233</td>
</tr>
<tr>
<td>s=ksubstr(str,1,1) ;</td>
<td>s=€</td>
</tr>
<tr>
<td>sl=klength(s) ;</td>
<td>l=4</td>
</tr>
<tr>
<td>l=klength(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>

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. UTF-8 is the only SAS session encoding supported by SAS Viya.

String functions are assigned I18N levels depending on whether the functions can process MBCS or SBCS data. 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 and MBCS (UTF-8) data.

**Table 10.4** 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. 310)</td>
<td>Returns a normalized string from an input string encoded in EBCDIC420.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“ANYALNUM Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for an alphanumeric character, and returns the first position at which the character is found.</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>“ANYALPHA Function” in SAS Viya Functions and CALL Routines: Reference</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 SAS Viya Functions and CALL Routines: Reference</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 SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for a digit, and returns the first position at which the digit is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYFIRST Function” in SAS Viya 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.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYGRAPH Function” in SAS Viya 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>X</td>
<td></td>
</tr>
<tr>
<td>“ANYLOWER Function” in SAS Viya 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>X</td>
<td></td>
</tr>
<tr>
<td>“ANYNAME Function” in SAS Viya 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.</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>“ANYPRINT Function” in SAS Viya 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 Viya 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>“ANYSPACE Function” in SAS Viya 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></td>
<td>X</td>
</tr>
<tr>
<td>“ANYUPPER Function” in SAS Viya 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></td>
<td>X</td>
</tr>
<tr>
<td>“ANYDIGIT Function” in SAS Viya 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.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“BASECHAR Function” (p. 314)</td>
<td>Converts characters to base characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“BYTE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns one character in the ASCII or the EBCDIC collating sequence.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“CAT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Does not remove leading or trailing blanks, and returns a concatenated character string.</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>“CATQ Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CATS Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Removes leading and trailing blanks, and returns a concatenated character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CATT Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Removes trailing blanks, and returns a concatenated character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CATX Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Removes leading and trailing blanks, inserts delimiters, and returns a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CHAR Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns a single character from a specified position in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CHOOSEC Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns a character value that represents the results of choosing from a list of arguments.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CHOOSEN Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns a numeric value that represents the results of choosing from a list of arguments.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COALESCEC Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns the first nonmissing value from a list of numeric arguments.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COLLATE Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns a character string in ASCII or EBCDIC collating sequence.</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>“COMPARE Function” in SAS Viya 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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPBL Function” in SAS Viya 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 Viya Functions and CALL Routines: Reference</td>
<td>Returns the generalized edit distance between two strings.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPLEV Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the Levenshtein edit distance between two strings.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPRESS Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a character string with specified characters removed from the original string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COUNT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Counts the number of times that a specified substring appears within a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COUNTC Function” in SAS Viya 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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COUNTW Function” in SAS Viya Functions and CALL Routines: Reference</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 SAS Viya Functions and CALL Routines: Reference</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>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>“FIND Function” in SAS Viya Functions and CALL Routines: Reference</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 SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a string for any character in a list of characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“FINDW Function” in SAS Viya Functions and CALL Routines: Reference</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 SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the first character in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“GETLOCENV Function” (p. 322)</td>
<td>Returns the current locale/language environment.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLANGUAGE Function” (p. 323)</td>
<td>Returns the current two-letter language code.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLOCALE Function” (p. 324)</td>
<td>Returns the POSIX locale value for a SAS locale.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXREGION Function” (p. 325)</td>
<td>Returns the current two-letter region code.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“HTMLDECODE Function” in SAS Viya 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></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“HTMLENCODE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Encodes characters using HTML character entity references, and returns the encoded string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“IFC Function” in SAS Viya 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></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>“IFN Function” in SAS Viya 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></td>
<td>X</td>
</tr>
<tr>
<td>“INDEX Function” in SAS Viya 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 Viya 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>“INDEXW Function” in SAS Viya Functions and CALL Routines: Reference</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></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KCOMPARE Function” (p. 326)</td>
<td>Returns the result of a comparison of character expressions.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KCOMPRESS Function” (p. 327)</td>
<td>Removes specified characters from a character expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KCOUNT Function” (p. 328)</td>
<td>Returns the number of double-byte characters in an expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KCVT Function” (p. 335)</td>
<td>Converts data from one type of encoding data to another type of encoding data.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KCOUNTC Function” (p. 329)</td>
<td>Counts the number of words in a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KCOUNTW Function” (p. 331)</td>
<td>Counts the number of words in a character string.</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>“KCOUNTX Function”</td>
<td>(p. 334) 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>“KFINDC Function”</td>
<td>(p. 339) Searches a string for any character in a list of characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KFINDX Function”</td>
<td>(p. 337) Searches for a specific substring of characters within a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEXC Function”</td>
<td>(p. 338) Searches a character expression for specified characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEX Function”</td>
<td>(p. 346) Searches a character expression for a string of characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEXB Function”</td>
<td>(p. 347) Searches a character expression for a string of characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEXCB Function”</td>
<td>(p. 349) Searches a character expression for specified characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLEFT Function”</td>
<td>(p. 350) Left-aligns a character expression by removing unnecessary leading DBCS blanks and Shift out/Shift in (SO-SI).</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLENGTH Function”</td>
<td>(p. 353) Returns the length of an argument.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLOWCASE Function”</td>
<td>(p. 354) Converts all letters in an argument to lowercase.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPCASE Function”</td>
<td>(p. 355) Converts Chinese, Japanese, Korean, Taiwanese (CJKT) 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>“KPROPCCHAR Function” (p. 358)</td>
<td>Converts special characters to normal characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPDATA Function” (p. 359)</td>
<td>Removes or converts unprintable characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KREVERSE Function” (p. 361)</td>
<td>Reverses a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KRIGHT Function” (p. 362)</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. 363)</td>
<td>Selects a specified word from a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSCANX Function” (p. 365)</td>
<td>Selects a specified word from a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSTRCAT Function” (p. 369)</td>
<td>Concatenates two or more character expressions.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSTRIP Function” (p. 370)</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. 372)</td>
<td>Extracts a substring from an argument.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSUBSTRN Function” (p. 373)</td>
<td>Returns a substring, allowing a result with a length of 0.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSUBSTRB Function” (p. 375)</td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KTRIM Function” (p. 377)</td>
<td>Removes trailing DBCS blanks and SO-SI from character expressions.</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;KTRUNCATE Function&quot; (p. 379)</td>
<td>Truncates a character string to a specified length in byte units without breaking multibyte characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;KUPCASE Function&quot; (p. 380)</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;KUPDATE Function&quot; (p. 381)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;KUPDATEB Function&quot; (p. 383)</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></td>
<td>X</td>
</tr>
<tr>
<td>&quot;KUPDATES Function&quot; (p. 384)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;KVERIFYB Function&quot; (p. 387)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;KVERIFY Function&quot; (p. 386)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;LEFT Function&quot; in SAS Viya Functions and CALL Routines: Reference</td>
<td>Left-aligns a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;LENGTH Function&quot; in SAS Viya 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></td>
<td>X</td>
</tr>
<tr>
<td>&quot;LENGTHC Function&quot; in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, including trailing blanks.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>&quot;LENGTHM Function&quot; in SAS Viya 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></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>“LENGTHN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, excluding trailing blanks.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“LOWCASE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to lowercase.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“MD5 Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the result of the message digest of a specified string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“MISSING Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a numeric result that indicates whether the argument contains a missing value.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“MVALID Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Checks the validity of a character string for use as a SAS member name.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLDATE Function” (p. 388)</td>
<td>Converts the SAS date value to the date value of the specified locale by using the date format descriptors.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLDATM Function” (p. 391)</td>
<td>Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLTIME Function” (p. 394)</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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLITERAL Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Converts a character string that you specify to a SAS name literal.</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>“NOTALNUM Function” in SAS Viya Functions and CALL Routines: Reference</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>X</td>
<td></td>
</tr>
<tr>
<td>“NOTALPHA Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for a non-alphabetic character, and returns the first position at which the character is found.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NOTCNTRL Function” in SAS Viya Functions and CALL Routines: Reference</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></td>
<td>X</td>
</tr>
<tr>
<td>“NOTDIGIT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for any character that is not a digit, and returns the first position at which that character is found.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NOTFIRST Function” in SAS Viya 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.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NOTGRAPH Function” in SAS Viya 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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NOTLOWER Function” in SAS Viya 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>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>“NOTNAME Function” in SAS Viya 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.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“NOTPRINT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for a non–printable character, and returns the first position at which that character is found.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NOTPUNCT Function” in SAS Viya 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>“NOTSPACE Function” in SAS Viya 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 Viya 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 Viya 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></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>“NVALID Function” in SAS Viya 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>“PROPCASE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Converts all words in an argument to proper case.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PRXCHANGE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Performs a pattern-matching replacement.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“PRXMATCH Function” in SAS Viya 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 Viya 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></td>
<td>X</td>
</tr>
<tr>
<td>“PRXPARSE Function” in SAS Viya 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></td>
<td>X</td>
</tr>
<tr>
<td>“PRXPOSN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a character string that contains the value for a capture buffer.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PUT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a value using a specified format.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PUTC Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Enables you to specify a character format at run time.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PUTN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Enables you to specify a numeric format at run time.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“QUOTE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Adds double quotation marks to a character value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“RANK Function” in SAS Viya 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 <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns a character value that consists of the first argument repeated ( n+1 ) times.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“REVERSE Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Reverses a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“RIGHT Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Right-aligns a character expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“SASMSG Function” (p. 395)</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></td>
<td>X</td>
</tr>
<tr>
<td>“SASMSGL Function” (p. 398)</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></td>
<td>X</td>
</tr>
<tr>
<td>“SCAN Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns the ( n )th word from a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SETLOCALE Function” (p. 404)</td>
<td>Specifies the locale keys for the current SAS locale.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“SHA256 Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns the result of the message digest of a specified string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“SORTKEY Function” (p. 401)</td>
<td>Creates a linguistic sort key.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“SOUNDEX Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Encodes a string to facilitate searching.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SPEDIS Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words.</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>“STRIP Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a character string with all leading and trailing blanks removed.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SUBPAD Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a substring that has a length that you specify, using blank padding if necessary.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SUBSTR (left of =) Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Replaces character value contents.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SUBSTR (right of =) Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Extracts a substring from an argument.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SUBSTRN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a substring, allowing a result with a length of 0.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRANSLATE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Replaces specific characters in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRANSTRN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Replaces or removes all occurrences of a substring in a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TRANTAB Function” on page 420</td>
<td>Transcodes data by using the specified translation table.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRANWRD Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Replaces or removes all occurrences of a substring in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRIM Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Removes trailing blanks from a character string, and returns one blank if the string is missing.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRIMN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Removes trailing blanks from character expressions, and returns a string with a length of 0 if the expression is missing.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONEDSTNAME Function” (p. 416)</td>
<td>Returns a daylight savings time name.</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>“TZONEDSTOFF Function”</td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 417)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONEID Function”</td>
<td>Returns the current time zone ID.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 411)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONENAME Function”</td>
<td>Returns the current standard or daylight savings time time zone name.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 412)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONEOFF Function”</td>
<td>Returns the user time zone offset.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 414)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONES2U Function”</td>
<td>Converts a SAS datetime value to a UTC datetime value.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 415)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONESTTNAME Function”</td>
<td>Returns a standard time zone name.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 418)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONESTTOFF Function”</td>
<td>Returns the time zone offset value for the specified standard time.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 419)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONEU2S Function”</td>
<td>Converts a UTC datetime value to a SAS datetime value.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 419)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UNICODE Function”</td>
<td>Converts Unicode characters to the current SAS session encoding.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 421)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UNICODEC Function”</td>
<td>Converts characters in the current SAS session encoding to Unicode characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 423)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UNICODELEN Function”</td>
<td>Specifies the length of the character unit for the Unicode data.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 424)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UNICODEWIDTH Function”</td>
<td>Specifies the length of a display unit for the Unicode data.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p. 425)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UPCASE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to uppercase.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></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>“URLDECODE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a string that was decoded using the URL escape syntax.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“URLENCODE Function” in SAS Viya 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 Viya 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>“WHICH Function” in SAS Viya Functions and CALL Routines: Reference</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>
Chapter 11

VARCHAR Support in Functions

VARCHAR Data Type in String Functions

SAS Viya supports the VARCHAR data type for character data. 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 `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.
The following examples compare the fixed-width data type and the VARCHAR data type. Table 11.1 on page 300 shows results from the fixed-width variable using the data €123. Table 11.2 on page 301 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.

Table 11.1 Fixed-Width Character Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; x='€123'; put x= x= $hex6.; xsub=substr(x,2); put xsub= xsub= $hex.; xidx=index(x,'1'); put xidx=; run;</td>
<td>x=€123 x=E282AC xsub='123' xsub=82AC31323320 xidx=4</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 $\text{vidx}=2$ because 1 is the second character in v.

Table 11.2  VARCHAR Character Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>length v VARCHAR(6);</td>
<td></td>
</tr>
<tr>
<td>v='€123';</td>
<td>v=€123 v=E282AC313233</td>
</tr>
<tr>
<td>put v=</td>
<td>vsub=123 vsub=31323320202020202020202020202020202020202020</td>
</tr>
<tr>
<td>v= $hex.;</td>
<td>vidx=2</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>
Chapter 12
Dictionary of NLS Call Routines and Functions

Functions by Category

Dictionary

ANORM420 Function ........................................ 310
BASECHAR Function ....................................... 314
CALL KSCANX Routine .................................... 316
ENCODCOMPAT Function .................................. 320
ENCODISVALID Function ................................. 321
GETLOCENV Function ..................................... 322
GETPXLANGUAGE Function .............................. 324
GETPXREGION Function ................................. 325
KCOMPARE Function ........................................ 326
KCOMPRESS Function ...................................... 327
KCOUNT Function ........................................... 328
KCOUNTC Function ......................................... 329
KCOUNTW Function ......................................... 331
KCOUNTX Function ......................................... 334
KCVT Function ............................................... 335
KFIND Function ............................................. 337
KFINDC Function ........................................... 339
KFINDW Function ........................................... 342
KINDEX Function ............................................ 346
KINDEXB Function .......................................... 347
KINDEXC Function .......................................... 348
KINDEXCB Function ........................................ 349
KLEFT Function ............................................... 350
KLENGTH Function .......................................... 353
KLOWCASE Function ....................................... 354
KPROPCHAR Function ...................................... 355
KPROPCHAR Function ...................................... 358
KPROPDATA Function ...................................... 359
KREVERSE Function ........................................ 361
KRIGH Function ............................................... 362
KSCAN Function ............................................. 363
KSCANX Function ........................................... 365
KSTRCAT Function .......................................... 369
KSTRIP Function ............................................. 370
KSUBSTR Function .......................................... 372
KSUBSTRN Function ........................................ 373
KSUBSTRB Function ........................................ 375
KTRANSLATE Function .................................... 376
Functions by Category

The following categories relate to NLS issues:

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.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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></td>
</tr>
<tr>
<td>DBCS</td>
<td></td>
</tr>
<tr>
<td>Date and Time</td>
<td></td>
</tr>
<tr>
<td>Locale</td>
<td></td>
</tr>
<tr>
<td>Variable Information</td>
<td></td>
</tr>
<tr>
<td><strong>CAS</strong></td>
<td><strong>BASECHAR Function (p. 314)</strong></td>
</tr>
<tr>
<td>CALL KSCANX Routine (p. 316)</td>
<td></td>
</tr>
<tr>
<td>KCOMPRESS Function (p. 327)</td>
<td></td>
</tr>
<tr>
<td>KCOUNT Function (p. 328)</td>
<td></td>
</tr>
<tr>
<td>KCOUNTC Function (p. 329)</td>
<td></td>
</tr>
<tr>
<td>KCOUNTW Function (p. 331)</td>
<td></td>
</tr>
<tr>
<td>KCOUNTX Function (p. 334)</td>
<td></td>
</tr>
<tr>
<td>KFIND Function (p. 337)</td>
<td></td>
</tr>
<tr>
<td>KFINDC Function (p. 339)</td>
<td></td>
</tr>
<tr>
<td>KFINDW Function (p. 342)</td>
<td></td>
</tr>
<tr>
<td>KINDEX Function (p. 346)</td>
<td></td>
</tr>
<tr>
<td>KINDEXC Function (p. 348)</td>
<td></td>
</tr>
<tr>
<td>KLEFT Function (p. 350)</td>
<td></td>
</tr>
<tr>
<td>KLENGTH Function (p. 353)</td>
<td></td>
</tr>
<tr>
<td>KLOWCASE Function (p. 354)</td>
<td></td>
</tr>
<tr>
<td>KREVERSE Function (p. 361)</td>
<td></td>
</tr>
<tr>
<td>KRIGHT Function (p. 362)</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>KSCAN Function (p. 363)</td>
</tr>
<tr>
<td></td>
<td>KSCANX Function (p. 365)</td>
</tr>
<tr>
<td></td>
<td>KSTRCAT Function (p. 369)</td>
</tr>
<tr>
<td></td>
<td>KSTRIP Function (p. 370)</td>
</tr>
<tr>
<td></td>
<td>KSUBSTR Function (p. 372)</td>
</tr>
<tr>
<td></td>
<td>KTRANSLATE Function (p. 376)</td>
</tr>
<tr>
<td></td>
<td>KTRIM Function (p. 377)</td>
</tr>
<tr>
<td></td>
<td>KUCASE Function (p. 380)</td>
</tr>
<tr>
<td></td>
<td>KUPDATE Function (p. 381)</td>
</tr>
<tr>
<td></td>
<td>KUPDATES Function (p. 384)</td>
</tr>
<tr>
<td></td>
<td>KVERIFY Function (p. 386)</td>
</tr>
<tr>
<td></td>
<td>VTRANSCODE Function (p. 427)</td>
</tr>
<tr>
<td></td>
<td>VTRANSCODEX Function (p. 428)</td>
</tr>
</tbody>
</table>

<p>| Character | ANORM420 Function (p. 310) | Returns a normalized string from an input string encoded in EBCDIC420. |
|          | BASECHAR Function (p. 314) | Converts characters to base characters. |
|          | KCVT Function (p. 335) | Converts data from one type of encoding data to another type of encoding data. |
|          | KSTRIP Function (p. 370) | Removes leading and trailing blanks from a character string. |
|          | TRANTAB Function (p. 420) | Transcodes data by using the specified translation table. |
|          | UNICODE Function (p. 421) | Converts Unicode characters to the current SAS session encoding. |
|          | UNICODEC Function (p. 423) | Converts characters in the current SAS session encoding to Unicode characters. |
|          | UNICODELEN Function (p. 424) | Specifies the length of the character unit for the Unicode data. |</p>
<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date and Time</td>
<td>UNICODEWIDTH Function (p. 425)</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. 388)</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>Date and Time</td>
<td>NLDATM Function (p. 391)</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>Date and Time</td>
<td>NLTIME Function (p. 394)</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>Date and Time</td>
<td>TZONEID Function (p. 411)</td>
<td>Returns the current time zone ID.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONENAME Function (p. 412)</td>
<td>Returns the current standard or daylight savings time, time zone name.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONEOFF Function (p. 414)</td>
<td>Returns the user time zone offset.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONE2S Function (p. 415)</td>
<td>Converts a SAS date time value to a UTC date time value.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONEDSTNAME Function (p. 416)</td>
<td>Returns a daylight savings time name.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONEDSTOFF Function (p. 417)</td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONESTTNAME Function (p. 418)</td>
<td>Returns a standard time zone name.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONESTTOFF Function (p. 419)</td>
<td>Returns the time zone offset value for the specified standard time.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>TZONEU2S Function (p. 419)</td>
<td>Converts a UTC date time value to a SAS date time value.</td>
</tr>
<tr>
<td>DBCS</td>
<td>CALL KSCANX Routine (p. 316)</td>
<td>Returns the position and length of the nth word from a character string.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOMPARE Function (p. 326)</td>
<td>Returns the result of a comparison of character expressions.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOMPRESS Function (p. 327)</td>
<td>Removes specified characters from a character expression.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOUNT Function (p. 328)</td>
<td>Returns the number of double-byte characters in an expression.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOUNTC Function (p. 329)</td>
<td>Counts the number of words in a character string.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOUNTW Function (p. 331)</td>
<td>Counts the number of words in a character string.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOUNTX Function (p. 334)</td>
<td>Counts the number of times that a specified substring appears within a character string.</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>KFIND Function (p. 337)</td>
<td>Searches for a specific substring of characters within a character string.</td>
</tr>
<tr>
<td></td>
<td>KFINDC Function (p. 339)</td>
<td>Searches a string for any character in a list of characters.</td>
</tr>
<tr>
<td></td>
<td>KFINDW Function (p. 342)</td>
<td>Returns the character position of a word in a string or returns the number of the word in a string.</td>
</tr>
<tr>
<td></td>
<td>KINDEX Function (p. 346)</td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td></td>
<td>KINDEXB Function (p. 347)</td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td></td>
<td>KINDEXC Function (p. 348)</td>
<td>Searches a character expression for specified characters.</td>
</tr>
<tr>
<td></td>
<td>KINDEXCB Function (p. 349)</td>
<td>Searches a character expression for specified characters.</td>
</tr>
<tr>
<td></td>
<td>KLEFT Function (p. 350)</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. 353)</td>
<td>Returns the length of an argument.</td>
</tr>
<tr>
<td></td>
<td>KLOWCASE Function (p. 354)</td>
<td>Converts all single-width English alphabet letters in an argument to lowercase.</td>
</tr>
<tr>
<td></td>
<td>KPROPCHAR Function (p. 358)</td>
<td>Converts special characters to normal characters.</td>
</tr>
<tr>
<td></td>
<td>KPROPDATA Function (p. 359)</td>
<td>Removes or converts unprintable characters.</td>
</tr>
<tr>
<td></td>
<td>KREVERSE Function (p. 361)</td>
<td>Reverses a character expression.</td>
</tr>
<tr>
<td></td>
<td>KRIGHT Function (p. 362)</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td></td>
<td>KSCAN Function (p. 363)</td>
<td>Selects a specified word from a character expression.</td>
</tr>
<tr>
<td></td>
<td>KSCANX Function (p. 365)</td>
<td>Selects a specified word from a character expression.</td>
</tr>
<tr>
<td></td>
<td>KSTRCAT Function (p. 369)</td>
<td>Concatenates two or more character expressions.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTR Function (p. 372)</td>
<td>Extracts a substring from an argument.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTRN Function (p. 373)</td>
<td>Returns a substring, allowing a result with a length of zero.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTRB Function (p. 375)</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>KTRANSLATE Function (p. 376)</td>
<td>Replaces specific characters in a character expression.</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><strong>KTRIM Function</strong> (p. 377)</td>
<td></td>
<td>Removes trailing DBCS blanks and SO/SI from character expressions.</td>
</tr>
<tr>
<td><strong>KTRUNCATE Function</strong> (p. 379)</td>
<td></td>
<td>Truncates a string to a specified length in byte unit without breaking multibyte characters.</td>
</tr>
<tr>
<td><strong>KUCASE</strong> Function (p. 380)</td>
<td></td>
<td>Converts all single-width English alphabet letters in an argument to uppercase.</td>
</tr>
<tr>
<td><strong>KUPDATE Function</strong> (p. 381)</td>
<td></td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td><strong>KUPDATEB Function</strong> (p. 383)</td>
<td></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><strong>KUPDATES Function</strong> (p. 384)</td>
<td></td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td><strong>KVERIFY Function</strong> (p. 386)</td>
<td></td>
<td>Returns the position of the first character that is unique to an expression.</td>
</tr>
<tr>
<td><strong>KVERIFYB Function</strong> (p. 387)</td>
<td></td>
<td>Returns the position of the first character that is unique to an expression.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ENCODCOMPAT Function</strong> (p. 320)</td>
<td></td>
<td>Verifies the transcoding compatibility between two encodings.</td>
</tr>
<tr>
<td><strong>ENCODEVALID Function</strong> (p. 321)</td>
<td></td>
<td>Verifies a valid encoding name.</td>
</tr>
<tr>
<td><strong>Locale</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GETLOCEnv Function</strong> (p. 322)</td>
<td></td>
<td>Returns the current locale/language environment.</td>
</tr>
<tr>
<td><strong>GETPXLANGUAGE Function</strong> (p. 323)</td>
<td></td>
<td>Returns the current two-letter language code.</td>
</tr>
<tr>
<td><strong>GETPXLOCALE Function</strong> (p. 324)</td>
<td></td>
<td>Returns the POSIX locale value for a SAS locale.</td>
</tr>
<tr>
<td><strong>GETPXREGION Function</strong> (p. 325)</td>
<td></td>
<td>Returns the current two-letter region code.</td>
</tr>
<tr>
<td><strong>SASMSG Function</strong> (p. 395)</td>
<td></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><strong>SASMSGL Function</strong> (p. 398)</td>
<td></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><strong>SORTKEY Function</strong> (p. 401)</td>
<td></td>
<td>Creates a linguistic sort key.</td>
</tr>
<tr>
<td><strong>SETLOCALE Function</strong> (p. 404)</td>
<td></td>
<td>Specifies the locale keys for the current SAS locale.</td>
</tr>
<tr>
<td><strong>Variable Information</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>VARTRANSCODE Function</strong> (p. 426)</td>
<td></td>
<td>Returns the transcode attribute of a SAS data set variable.</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>VTRANSCODE Function (p. 427)</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. 428)</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 279](#).

### Syntax

\[ \text{ANORM420} (\text{string},<\text{modifiers}>) \]

### 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 12.2  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>ﱐ</td>
<td>(U+FEF5)</td>
<td>0xB2</td>
<td>0xB1 + 0x47</td>
<td>Arabic Ligature Lam with Aleph with Madda above - Isolated form</td>
</tr>
<tr>
<td>ﱑ</td>
<td>(U+FEF6)</td>
<td>0xB3</td>
<td>0xB1 + 0x47</td>
<td>Arabic Ligature Lam with Aleph with Madda above - Final form</td>
</tr>
<tr>
<td>ﱒ</td>
<td>(U+FEF7)</td>
<td>0xB4</td>
<td>0xB1 + 0x49</td>
<td>Arabic Ligature Lam with Aleph with Hamza above - Isolated form</td>
</tr>
<tr>
<td>ﱓ</td>
<td>(U+FEF8)</td>
<td>0xB5</td>
<td>0xB1 + 0x49</td>
<td>Arabic Ligature Lam with Aleph with Hamza above - Final form</td>
</tr>
<tr>
<td>ﱔ</td>
<td>(U+FEFC)</td>
<td>0xB8</td>
<td>0xB1 + 0x56</td>
<td>Arabic Ligature Lam with Aleph - Isolated form</td>
</tr>
<tr>
<td>ﱕ</td>
<td>(U+FEFD)</td>
<td>0xB9</td>
<td>0xB1 + 0x56</td>
<td>Arabic Ligature Lam with Aleph - Final form</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 12.3  Modifier \( i \): Arabic-Indic Numbers**

<table>
<thead>
<tr>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>٠ U+0660</td>
<td>٠</td>
<td>١</td>
<td>١</td>
<td>٣</td>
<td>٣</td>
<td>٥</td>
<td>٥</td>
<td>٧</td>
<td>٧</td>
<td>٩</td>
</tr>
<tr>
<td>١ U+0661</td>
<td>١</td>
<td>٢</td>
<td>٢</td>
<td>٤</td>
<td>٤</td>
<td>٦</td>
<td>٦</td>
<td>٨</td>
<td>٨</td>
<td></td>
</tr>
<tr>
<td>٢ U+0662</td>
<td></td>
<td>٣</td>
<td>٣</td>
<td>٥</td>
<td>٥</td>
<td>٧</td>
<td>٧</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>٣ U+0663</td>
<td></td>
<td></td>
<td>٤</td>
<td>٤</td>
<td>٦</td>
<td>٦</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>٤ U+0664</td>
<td></td>
<td></td>
<td></td>
<td>٥</td>
<td>٥</td>
<td>٧</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>٥ U+0665</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>٦</td>
<td>٦</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>٦ U+0666</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>٧</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>٧ U+0667</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>٨</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>٨ U+0668</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>٩</td>
<td></td>
<td></td>
</tr>
<tr>
<td>٩ U+0669</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>٠</td>
<td></td>
</tr>
</tbody>
</table>

Hexadecimal: 0xdf 0xea 0xeb 0xed 0xee 0xef 0xfb 0xfc 0xfd 0xFE

To: 0 1 2 3 4 5 6 7 8 9

Hexadecimal: 0xf0 0xf1 0xf2 0xf3 0xf4 0xf5 0xf6 0xf7 0xf8 0xF9

The ANORM420 function remaps the shaped characters to their unshaped equivalent unless the modifier \( p \) is specified.

**Table 12.4  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></td>
<td>0x42</td>
</tr>
<tr>
<td>0x48</td>
<td>FE82</td>
</tr>
<tr>
<td></td>
<td>0x47</td>
</tr>
<tr>
<td></td>
<td>U+0622</td>
</tr>
<tr>
<td>From</td>
<td>To</td>
</tr>
<tr>
<td>--------</td>
<td>----------</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>
<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>
</tbody>
</table>
The ANORM420 function adds a space after the following characters unless the modifier is specified.

### Table 12.5 Modifier s: ignores the addition of a space

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Arabic Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xCD</td>
<td>U+FEEC</td>
<td>U+0647</td>
</tr>
<tr>
<td>0xDB</td>
<td>U+FEF0</td>
<td>U+061F</td>
</tr>
<tr>
<td>0xDD</td>
<td>U+FEF2</td>
<td>U+064A</td>
</tr>
<tr>
<td>0xDE</td>
<td>U+FEF3</td>
<td>U+064A</td>
</tr>
</tbody>
</table>

| ب     | 0x58    | (U+0628) BEH |
| ة     | 0x62    | (U+0629) TEH MARBUTA |
| ت     | 0x63    | (U+062A) TEH |
| ث     | 0x65    | (U+062B) THEH |
| ج     | 0x67    | (U+062C) JEEM |
| ح     | 0x69    | (U+062D) HAH |
| خ     | 0x71    | (U+062E) KHAH |
| س     | 0x77    | (U+0633) SEEN |
| ش     | 0x80    | (U+0634) SHEEN |
| ص     | 0x8B    | (U+0635) SAD |
| ض     | 0x8D    | (U+0636) DAD |
| ع     | 0x9A    | (U+0639) AIN |
| ف     | 0x9B    | (U+FECA) AIN final form |
| غ     | 0x9E    | (U+063A) GHAIN |
| خ     | 0x9F    | (U+FECE) GHAIN final form |
| ف     | 0xAB    | (U+0641) FEH |
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**

The following example demonstrates the ANORM420 function:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>s1=C8E5C7E3C7D320A02020</td>
</tr>
<tr>
<td>a = '59CD57BC577745'x ;</td>
<td>s2=C8E5C7E3C7D3A0202020</td>
</tr>
<tr>
<td>s1 = anorm420(a) ;</td>
<td>s3=59CD57BC577740454040</td>
</tr>
<tr>
<td>/* Turn off addition of space and mapping of Arabic-Indic numbers */</td>
<td></td>
</tr>
<tr>
<td>s2 = anorm420(a,&quot;si&quot;) ;</td>
<td></td>
</tr>
<tr>
<td>/* Turn off transcoding */</td>
<td></td>
</tr>
<tr>
<td>s3 = anorm420(a,'t') ;</td>
<td></td>
</tr>
<tr>
<td>put s1= $hex20. / s2= $hex20. / s3= $hex20. ;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**BASECHAR Function**

Converts characters to base characters.
Syntax

\[
\text{STR=BASECHAR}(\text{<instr>}, (<\text{Unicode type}>))
\]

Required Arguments

\textit{str}
- data string that is converted.

\textit{instr}
- input data string.

\textit{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 following examples demonstrate using the Unicode types:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data cities; length name$24 name_utf8$24; input name; name_utf8 = basechar(name); put name_utf8=; cards; Mühlenfließ LaUña ZielonaGóra run;</td>
<td>Note: The BASECHAR function uses Unicode character decomposition mappings, so the sharp s, ß, is valid. <a href="http://www.unicode.org/reports/tr44/#Character_Decomposition_Mappings">http://www.unicode.org/reports/tr44/#Character_Decomposition_Mappings</a> Muhlenfließ LaUna ZielonaGora</td>
</tr>
</tbody>
</table>
**Statements**

Note: The SAS session encoding is UTF-8 and the ESC option is specified.

data cities;length name$24 name_utf8$80;
  input name;
  name_utf8 = basechar(name, "ESC"); put name_utf8=;
cards;
Mühlenfließ
LaUna
ZielonaGóra
run;

Note: The SAS session encoding is UTF-8 and the NCR option is specified.

data cities;length name$24 name_utf8$80;
  input name;
  name_utf8 = basechar(name, "NCR"); put name_utf8=;
cards;
Mühlenfließ
LaUna
ZielonaGóra
run;

---

**CALL KSCANX Routine**

Returns the position and length of the $n$th word from a character string.

**Categories:** DBCS CAS

**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 on page 279.

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 log.

**Syntax**

```
CALL KSCANX(<string>,count,position,length <,<character-list><,<modifier(s)>>>);
```

**Required Arguments**

- `string` specifies a character constant, variable, or expression.

- `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 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, SCAN counts words from left to right in the character string.
• If `count` is negative, SCAN 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 zero. If `count` is zero 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 zero. If `count` is zero 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 SCAN 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 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 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 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 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.

- **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.

**Tip** If the modifier argument is a character constant, then enclose it 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**

**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.

If you specify the Q modifier, then delimiters inside substrings that are enclosed in
quotation marks are ignored.

In the CALL KSCANX routine, 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. However, the CALL
KSCANX routine 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 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 as follows:
  - 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 ! $ % & ( ) * + , - . / ; ¬ | ç

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 zero 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 zero.

**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 zero.

**Finding the Word as a Character String**

To find the designated word as a character string after calling the CALL KSCANX routine, use the SUBSTRN function with the `string`, `position`, and `length` arguments:
Because CALL KSCANX can return a length of zero, using the SUBSTR function can cause an error.

**Using Null Arguments**
The CALL KSCANX routine allows character arguments to be null. Null arguments are treated as character strings with a length of zero. Numeric arguments cannot be null.

**Example**

This example uses Chinese characters:

```plaintext
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。'
  call kscanx(xyz, 2, pos, len, ' ', 'sp'); /* Use spaces and punctuations as delimiter. */
  put pos= len=
run;
```

Here is the result from the CALL KSCANX routine example:

```
pos=17 len=3
```

**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 \( \text{Source1} \) is not a valid encoding name.
-2 \( \text{Source2} \) 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 following examples demonstrate the ENCODCOMPAT features:

| Statements                                                                 | Results |
|/etc session encoding is wlatin1 */ isCompat= EncodCompat("xyz"); put isCompat;  | -1      |
| /* session encoding is wlatin1 */ isCompat= EncodCompat ("cp1252"); put isCompat; | 1       |
| isCompat= EncodCompat ("ebcdic1149","open_ed-1149"); put isCompat;             | 2       |
| isCompat= EncodCompat ("cp1251","ebcdic1149"); put isCompat;                  | 0       |

**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.
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

The following examples demonstrate the ENCODISVALID features:

<table>
<thead>
<tr>
<th>SAS Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>isValid= EncodIsValid(&quot;xyz&quot;); put isValid;</td>
<td>0</td>
</tr>
<tr>
<td>isValid= EncodIsValid(&quot;wlt2&quot;); put isValid;</td>
<td>1</td>
</tr>
<tr>
<td>isValid= EncodIsValid(&quot;wlatin2&quot;); put isValid;</td>
<td>2</td>
</tr>
<tr>
<td>isValid= EncodIsValid(&quot;cp1250&quot;); put isValid;</td>
<td>3</td>
</tr>
</tbody>
</table>

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 279.

**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.
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
In the following example, the LOCALE= system option is set to French_France.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france;</td>
<td></td>
</tr>
<tr>
<td>environ=getlocenv();</td>
<td>SBCS</td>
</tr>
<tr>
<td>put environ;</td>
<td></td>
</tr>
</tbody>
</table>

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 279.

**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>
</tbody>
</table>
option locale=German;
lang=getpxLanguage();
put lang;

option locale=en_US;
lang=getpxLanguage();
put lang;

See Also

System Options:
•  “LOCALE System Option” on page 578

Functions:
•  “GETPXREGION Function” on page 325
•  “GETPXLOCALE Function” on page 324

GETPXLOCALE Function

Returns the POSIX locale value for a 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 279.

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>
<tr>
<td>locale=getpxLocale(&quot;german_germany&quot;);</td>
<td>de_DE</td>
</tr>
<tr>
<td>put locale;</td>
<td></td>
</tr>
<tr>
<td>locale=getpxLocale(&quot;english_unitedstates&quot;);</td>
<td>en_US</td>
</tr>
<tr>
<td>put locale;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**System Options:**
- “LOCALE System Option” on page 578

**Functions:**
- “GETPXLANGUAGE Function” on page 323
- “GETPXREGION Function” on page 325

---

### 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 279](#).

#### 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 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>FR</td>
</tr>
<tr>
<td>region=getpxRegion(); put region;</td>
<td></td>
</tr>
<tr>
<td>option locale=German;</td>
<td>DE</td>
</tr>
<tr>
<td>region=getpxRegion(); put region;</td>
<td></td>
</tr>
<tr>
<td>option locale=en_US;</td>
<td>US</td>
</tr>
<tr>
<td>region=getpxRegion(); put region;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

System Options:
- “LOCALE System Option” on page 578

Functions:
- “GETPXLOCALE Function” on page 324
- “GETPXLANGUAGE Function” on page 323

KCOMPARE Function

Returns the result of a comparison of character expressions.

**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 279.

**Tip:** Non-DBCS equivalent function is the “COMPARE Function” in SAS Viya Functions and CALL Routines: Reference.

**Syntax**

\[ \text{KCOMPARE}( \text{source}, \text{<pos, <count,> findstr} ) \]
**Required Arguments**

*source*

specifies the character expression to be compared.

*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.

*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.

*findstr*

specifies the character expression to compare to *source*.

**Details**

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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options page no=1 nodate ls=80 ps=80;</td>
<td>The SAS System</td>
</tr>
<tr>
<td>data test;</td>
<td></td>
</tr>
<tr>
<td>rc1 = kcompare(’漢字のテスト’, ’汉字’);</td>
<td>0 3 0 3 1 4</td>
</tr>
<tr>
<td>rc2 = kcompare(’漢字のテスト’, 7, ’テスト’);</td>
<td></td>
</tr>
<tr>
<td>rc3 = kcompare(’漢字のテスト’, 1, 8, ’汉字’);</td>
<td></td>
</tr>
<tr>
<td>rc4 = kcompare(’漢字のテスト’, ’ABC’);</td>
<td></td>
</tr>
<tr>
<td>rc5 = kcompare(’ABCDEF’, ’ABC’);</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>proc print data=test; run;</td>
<td></td>
</tr>
</tbody>
</table>

**KCOMPARE Function**

Removes specified characters from a character expression.

**Categories:** DBCS, CAS
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 279.

Note: This function supports the varchar type.

Tip: Non-DBCS equivalent function is COMPRESS in SAS Viya Functions and CALL Routines: Reference.

Syntax

KCOMPRESS(source,<characters-to-remove>)

Required Arguments

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.

characters-to-remove specifies the character or characters that KCOMPRESS removes from the character. If characters-to-remove is omitted, KCOMPRESS removes all blanks.

Tip Enclose a literal string of characters in quotation marks.

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>; result=kcompress('漢字のテスト',')'; put result=; run;</td>
<td>result=漢字テスト</td>
</tr>
</tbody>
</table>

See Also

Functions:

- “KLEFT Function” on page 350
- “KTRIM Function” on page 377

KCOUNT Function

Returns the number of double-byte characters in an expression.

Categories: DBCS CAS
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 279.

Note: This function supports the varchar type.

Syntax

\[ \text{KCOUNT}(\text{source}) \]

Required Argument

\textit{source}

specifies the character expression to count.

Details

See “Internationalization Compatibility for SAS String Functions” on page 279 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>; text='abcあいう'; result=kcount(text); put result=; run;</td>
<td>result=3</td>
</tr>
<tr>
<td>data casuser.encodcompat; text='abcあいう'; result=kcount(text); put result=; run;</td>
<td>result=3</td>
</tr>
</tbody>
</table>

**KCOUNTC Function**

Counts the number of words in a character string.

**Categories:**

- DBCS
- CAS

**Restriction:**

This function is assigned an I18N Level 2 status and is designed for DBCS data. However, if the first argument, \textit{string}, has single-byte characters, then the KCOUNTC function processes the SBCS data. For more information, see Internationalization Compatibility on page 279.
Syntax

KCOUNTC(string, character-list <, modifier(s)>)

Required Arguments

string
specifies a character constant, variable, or expression in which characters are counted.

Tip  Enclose a literal string of characters in quotation marks.

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.

Tips  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

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:

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.
f or F  adds an underscore and English letters (that is, the characters that can begin a SAS variable name using VALIDVARNAME=V7) to the list of characters.
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 case.
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 string and character-list. If you want to remove trailing blanks from only one character argument instead of both (or all) character arguments, use the TRIM function instead of the KCOUNTC function 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 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:

```
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=kcountc(xyz, '软件'); /* Count '软件' occurrence */
  put howmanythis = ;
run;
```

Here is the result from the KCOUNTC function example:

```
howmanythis=4
```

KCOUNTW Function

Counts the number of words in a character string.

Categories:  DBCS
**CAS**

**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 on page 279](#).

**Syntax**

\[
\texttt{KCOUNTW} (<\textit{string}>, <\textit{character-list}>, <\textit{modifier}>)
\]

**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, the characters that can begin a SAS variable name using `VALIDVARNAMES=V7`) to the list of characters.
- 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.
- 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 after the first character in a SAS variable name using VALIDVARNAMES=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 (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed) to the list of characters.

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 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;
```

Here is the result from the KCOUNTW function example:

```
howmanythis=2
```

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

**Categories:**
- DBCS
- CAS

**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 KCOUNTX function processes the SBCS data. For more information, see Internationalization Compatibility on page 279.

**Syntax**

```
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 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 as an expression.

### Details

#### The 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, \texttt{count(‘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=kcountx(xyz, ‘软件’); /* Count ‘软件’ occurrence */
  put howmanythis = ;
run;
```

Here is the result from the KCOUNTX function example:

```
howmanythis=2
```

### KCVT Function

Converts data from one type of encoding data to another type of encoding data.

**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 279.
Syntax

\[ \text{KCVT}(\text{text, intype, outtype, <options, ...>}) \]

**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 643.
  
  ASCIIANY and EBCIDICANY 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 643.
  
  ASCIIANY and EBCIDICANY 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 279 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.

**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.

```plaintext
data _null_;  
   infile 'my-input-file';  
   file outdd noprint;  
   input @1 text $char80.;  
   text = kcvt(text, 'pcibm', 'dec');  
   put @1 text $char80.;  
run;
```

---

**KFINDFunction**

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

**Categories:** DBCS  
CAS

**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 KFINDFunction processes the multi-byte data. For more information, see Internationalization Compatibility on page 279

**Syntax**

```
KFINDF(*string*, substring <start-position> <modifier(s)>)
```

**Required Arguments**

*string*

specifies a character constant, variable, or expression that is searched for substrings.

**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 search for in *string*.

**Tip** Enclose a literal string of characters in quotation marks.

**Optional Arguments**

*modifier(s)*

is a character constant, variable, or expression that specifies one or more modifiers.

The following modifiers are valid:

* 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 TRIM 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**

**The 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 <em>startpos</em></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:

```plaintext
data _null_;
xz='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=kfind(xz, '软件'); /* Find first position of '软件' */
put pos_of = ;
run;
```

Here is the result from the KFIND function example:

```
pos_of=10
```
**KFINDC Function**

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

**Categories:** DBCS  
CAS

**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 on page 279](#).

**Note:** This function supports the varchar type.

### Syntax

\[
\text{KFINDC}(\text{string} <,\text{character-list}>) \\
\text{KFINDC}(\text{string},\text{character-list}<,\text{modifier(s)}> \\
\text{KFINDC}(\text{string},\text{character-list},\text{modifier(s)}<,\text{start-position}> \\
\text{KFINDC}(\text{string},\text{character-list},<\text{start-position}>,<\text{modifier(s)}>)
\]

### 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.
  
  - **a** or **A**  
  adds alphabetic characters to the list of characters.
  
  - **b** or **B**  
  searches from right to left instead of from left to right, regardless of the sign of the **start-position** 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, the characters that can begin a
   SAS variable name using VALIDVARNAME=V7) to the list of characters.

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 character case during the search.

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.

   Note: If you want to remove trailing blanks from just one character argument
   instead of both (or all) character arguments, use the TRIM function instead of
   the KFINDC function 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 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**

**The 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**

Here is an example of the KFINDC function:

```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=kfindc(xyz, '软件的'); /* Find minimum position of '软', '件' or '的' */
put pos_of = ;
run;
```

Here is the result from the KFINDC function example:
KFINDW Function

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

**Categories:** DBCS

**CAS**

**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 on page 279](#).

**Syntax**

KFINDW(*string*,*word* [*<character-list>*])

KFINDW(*string*,*word*,*character-list*,*modifier*[*<start-position>*])

KFINDW(*string*,*word*,*character-list*,*start-position*,*modifier*)

KFINDW(*string*,*word*,*start-position*,*character-list*,*modifier*)

**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 the following characters as modifiers:
blank
is ignored.
a or A
adds alphabetic characters to the list of characters.
b or B
scans from right to left instead of from left to right, regardless of the sign of the
start-position argument.
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, the characters that can begin a
SAS variable name using VALIDVARNAME=V7) to the list of characters.
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.
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.
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.
n or N
adds digits, an underscore, and English letters (that is, the characters that can
appear after the first character 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 (blank, horizontal tab, vertical tab, carriage return, line feed, and form feed) to the list of characters.

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 characters to the list of characters.

x or X
adds hexadecimal characters to the list of characters.

Details

Definition of 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.

Definition of 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 startpos</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than 0</td>
<td>Search begins at position start-position and proceeds to the right. If start-position 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 start-position and proceeds to the left. If start-position 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.

**Example**

Here is an example of the KFINDW function:

```plaintext
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。";  
pos_of=kfindw(xyz, '软件'); /* The default delimiter is SBCS blank. */  
put pos_of = ;  
run;
```
Here is the result from the KFINDW function example:

```
pos_of=11
```

Here is another example of the KFINDW function:

```r
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。';
pos_of=kfindw(xyz,'sas',',','sp'); /* Use spaces and punctuations as delimiter. */
put pos_of =;
run;
```

Here is the result from the KFINDW function example:

```
pos_of=17
```

---

**KINDEX Function**

Searches a character expression for a string of characters.

**Categories:** DBCS, CAS

**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 279.

**Note:** This function supports the varchar type.

**Tip:** Non-DBCS equivalent function is INDEX in SAS Viya 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 279 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, the KINDEX function returns a value of 0. If there are multiple occurrences of the string, the KINDEX function 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>
<tbody>
<tr>
<td>data <em>null</em>; 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;</td>
<td>result=7 result=0</td>
</tr>
</tbody>
</table>

See Also

Functions:

- “KINDEXC Function” on page 348

KINDEXB Function

Searches a character expression for a string of 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 279.

**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. Most Latin characters are 2 bytes wide and CJK characters are 3 bytes wide, that makes these functions 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.

```
data null;
  text='漢字文字列の検索';
  result=kindexb(text, '検索');
  put result=;
  text2='探

  result=kindexb(text, text2);
  put result=;
run;
```

KINDEXCB Function

Searches a character expression for specified characters.

Categories: DBCS  CAS

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 279.

Note: This function supports the varchar type.

Tip: The Non-DBCS equivalent function is "INDEXC Function" in SAS Viya Functions and CALL Routines: Reference

Syntax

KINDEXC(source, excerpt-1, ...excerpt-n)
**Required Arguments**

*source*

specifies the character expression to search for.

*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 279 for restrictions and more information.

The KINDEXC function searches `source`, from left to right, for the first occurrence of any character present in the excerpts and returns the position in `source` of that character. If none of the characters in `excerpt-1` through `excerpt-n` in `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 346

**KINDEXCB Function**

Searches a character expression for specified 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 279.

Syntax

KINDEXCB(source, excerpt-1, ...excerpt-n)

Required Arguments

source
specifies the character expression to search for.

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

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. Most Latin characters are 2 bytes wide and CJK characters are 3 bytes wide, that makes these functions 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>a='A B C def (漢字)123456';</td>
<td></td>
</tr>
<tr>
<td>result=kindexcb(a, '1234567880’, ’感潰’);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>Result=19</td>
<td></td>
</tr>
</tbody>
</table>

KLEFT Function

Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.
Categories: DBCS
CAS

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 279.

Note: This function supports the varchar type.

Tip: Non-DBCS equivalent function is LEFT in SAS Viya 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 279 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 12.6 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>U+0020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+0020</td>
<td>SINGLE-BYTE SPACE, DEPEND ON COMPILING</td>
<td></td>
</tr>
<tr>
<td>U+0085</td>
<td>NEXT LINE</td>
<td></td>
</tr>
<tr>
<td>U+00A0</td>
<td>NO-BREAK SPACE</td>
<td></td>
</tr>
<tr>
<td>U+1680</td>
<td>OGHAM SPACE MARK</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+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 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.
KLENGTH Function

Returns the length of an argument.

**Categories:**
- DBCS
- CAS

**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 279.

**Note:**
This function supports the varchar type.

**Tip:**
Non-DBCS equivalent function is LENGTH in SAS Viya Functions and CALL Routines: Reference.

**Syntax**

\[
\text{KLENGTH}(\text{argument})
\]

**Required Argument**

\*argument\*

specifies any SAS expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 279 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 English VARCHAR 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));
```

This code improves performance and the results reflect the LENGTH 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>; text='abcあいう'; result=klength(text); put result=; run;</td>
<td>result=6</td>
</tr>
</tbody>
</table>

### KLOWCASE Function

Converts all single-width English alphabet letters in an argument to lowercase.

**Categories:** DBCS

**CAS**

**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 279](#).

**Note:** This function supports the varchar type.

**Tip:** Non-DBCS equivalent function is LOWCASE in [SAS Viya Functions and CALL Routines: Reference](#).

**Syntax**

\[ KLOWCASE(argument) \]

**Required Argument**

\[ argument \]

specifies any SAS character expression.
Details

See “Internationalization Compatibility for SAS String Functions” on page 279 for restrictions and more information.

The KLOWCASE function copies a character argument, converts all uppercase single-width English alphabet letters to lowercase letters, and returns the altered value as a result.

Example

The following example uses Japanese characters.

```sas
data _null_;  
result=klowcase('ABCあいうえお');  
put result=;  
run;
```

KPROPCEASE 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 279.

Syntax

```
str=KPROPCEASE(<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:
  • Full-alphabet, Half-alphabet

Details

See “Internationalization Compatibility for SAS String Functions” on page 279 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:

```
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
```
KPROPCHAR Function

Converts special characters to normal 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 279.

**Syntax**

\[
str = \text{KPROPCHAR}(\text{<instr}>)
\]

**Required Arguments**

- **str**
  
  result string. Special characters are converted to normal characters.

- **instr**
  
  input data string.

**Details**

This function converts special characters to normal characters. The KPROPCHAR function converts the characters from the following ranges:


**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)
```
### Statements

```plaintext
data _null_;  
length in1 out1 $30 ;  
in1=unicode('①⑳❿㉟');  
out1=KPROPCHAR(in1);  
put out1;  
run;
```

### Results

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(20)</td>
<td>(-10)</td>
<td>(35)</td>
</tr>
</tbody>
</table>

```plaintext
data _null_;  
length in1 out1 $30 ;  
in1=unicode('❶➓');  
out1=KPROPCHAR(in1);  
put out1;  
run;
```

### Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(-1)</td>
<td>(-10)</td>
</tr>
</tbody>
</table>

```plaintext
data _null_;  
length in1 out1 $30 ;  
in1=unicode('㈀㋾');  
out1=KPROPCHAR(in1);  
put out1;  
run;
```

### Results

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(utive)</td>
<td>(ive)</td>
<td>(ive)</td>
<td></td>
</tr>
</tbody>
</table>

---

### 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 279.

### Syntax

```plaintext
str=KPROPDATA(<instr> (<options>, <input encode name>, <output encode name>))
```

### Required Arguments

- **str**
  data string that has been converted and is in session encoding.

- **instr**
  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.

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 643.

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 643.

Details
This function converts the input data string to the current SAS session encoding and removes or replaces unprintable characters based on the options.

Example
The following example demonstrates the functionality of the KPROPDATA function:

```
length instr $12;
length str1 str2 str3 str4 str5 str6 str7 str8 str9 str10$ 50;
instr = "534153"x||"ae"x || " System";
pot instr;
str1 = kpropdata(instr);
pot str1= +2 str1= $hex26.;
str2 = kpropdata(instr,'UESC');
pot str2= +2 str2= $hex26.;
str3 = kpropdata(instr, 'UESC','wlatin1');
pot str3= +2 str3= $hex34.;
str4 = kpropdata(instr,'TRIM','wlatin1');
pot str4= +2 str4= $hex26.;
str5 = kpropdata(instr,'BLANK', 'wlatin1');
pot str5= +2 str5= $hex26.;
str6 = kpropdata(instr,','?', 'wlatin1');
pot str6= +2 str6= $hex26.;
str7 = kpropdata(instr,'hex', 'wlatin1');
pot str7= +2 str7= $hex26.;
str8 = kpropdata(instr,'TRUNC', 'wlatin1');
```
KREVERSE Function

Reverses a character expression.

Categories:  DBCS
CAS

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 279.

Note:  This function supports the varchar type.

Tip:  Non-DBCS equivalent function is REVERSE in SAS Viya 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 279 for restrictions and more information.

Example

The following example uses Japanese characters.
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=トステの字漢</td>
</tr>
<tr>
<td>result=k.reverse('漢字のテスト');</td>
<td></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:**  
- DBCS  
- CAS

**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 279.

**Note:**  
This function supports the varchar type.

**Tip:**  
See “RIGHT Function” in SAS Viya Functions and CALL Routines: Reference.

### Syntax

KRIGHT(argument)

### Required Argument

**argument**  
specifies any SAS character expression.

### Details

See “Internationalization Compatibility for SAS String Functions” on page 279 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 12.6 on page 351 for a list of the Unicode characters.

### Example

The following example uses Japanese characters.
KSCAN Function

Selects a specified word from a character expression.

**Categories:** DBCS, CAS

**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 279.

**Note:** This function supports the varchar type.

**Tip:** Non-DBCS equivalent function is SCAN in SAS Viya 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.

---

**Statements**

```sas
data _null_;
  a='漢字のテスト';
  b=kright(a);
  put '-----1------2-----';
  put a $25.;
  put b $25.;
run;
```

---

**See Also**

**Functions:**
- “KCOMPRESS Function” on page 327
- “KLEFT Function” on page 350
- “KTRIM Function” on page 377

---

KSCAN Function

Selects a specified word from a character expression.

**Categories:** DBCS, CAS

**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 279.

**Note:** This function supports the varchar type.

**Tip:** Non-DBCS equivalent function is SCAN in SAS Viya 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 279 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.

**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>length x $20 y $20;</td>
<td></td>
</tr>
<tr>
<td>text1='これは漢字関数のテストです。';</td>
<td>x=これ</td>
</tr>
<tr>
<td>x='nonblank';</td>
<td>x=漢字関数</td>
</tr>
<tr>
<td>y='nonblank';</td>
<td>x=テストです。</td>
</tr>
<tr>
<td>do until(x=' ');</td>
<td></td>
</tr>
<tr>
<td>x=kscan(text1,i,'はの');</td>
<td>x=</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>y=             ;</td>
<td></td>
</tr>
<tr>
<td>end;</td>
<td></td>
</tr>
<tr>
<td>y='nonblank';</td>
<td></td>
</tr>
<tr>
<td>i=-1;</td>
<td></td>
</tr>
<tr>
<td>do until(y=' ');</td>
<td></td>
</tr>
<tr>
<td>y=kscan(text1,i,'はの');</td>
<td>y=</td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>i=i-1;</td>
<td></td>
</tr>
<tr>
<td>end;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
KSCANX Function

Selects a specified word from a character expression.

**Categories:**
- DBCS
- CAS

**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 on page 279.

**Syntax**

```plaintext
KSCANX(string, count <, character-list <, modifier>>)
```

**Required Arguments**

- `string`
  - specifies a character constant, variable, or expression.

- `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**

- `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.
    - Specifying a modifier can also change the character-list used as delimiters. For example, if you specify the K modifier in the `modifier` argument, then all characters that are not in `character-list` are used as delimiters.

  **Note:** For more information see: “Using Default Delimiters in ASCII and EBCDIC Environments” in SAS Viya Functions and CALL Routines: Reference.

- `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:

**Tip** You can add more characters to `character-list` by using other 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 by using VALIDVARNAME=V7) to the list of characters.
g or G adds graphic characters to the list of characters. Graphic characters are characters 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 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, 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 by using VALIDVARNAME=V7) to the list of characters.
o or O processes the charlist 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 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 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 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 charlist arguments. If you want to remove trailing blanks from only one character argument instead of both character arguments, use the TRIM function instead of the KSCANX function with the T modifier.

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.

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

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 charlist 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
- 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. 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 as follows:
  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:
  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 charlist argument. Modifiers do not add to the list of default modifiers.
The Length of the Result
In a DATA step, most variables have a fixed length. If the word returned by the KSCANX function is assigned to a variable that has a fixed length greater than the length of the returned word, then the value of that variable is padded with blanks. Macro variables have varying lengths and are not padded with blanks.

The maximum length of the word that is returned by the KSCANX function depends on the environment from which it is called:

• In a DATA step, if the KSCANX function returns a value to a variable that has not yet been given a length, that variable is given the length of the first argument. This behavior is different from the behavior in previous releases of SAS. In previous releases code that created a variable with a length of 200 might have produced a variable with a length that was greater than expected. If you need the KSCANX function to assign to a variable a value that is different from the length of the first argument, use a LENGTH statement for that variable before the statement that uses the KSCANX function.

If you use the KSCANX function in an expression that contains operators or other functions, a word that is returned by the KSCANX function can have a length of up to 32,767 characters, except in a WHERE clause. In that case, the maximum length is 200 characters.

• In the SQL procedure, or in a WHERE clause in any procedure, the maximum length of a word that is returned by the KSCANX function is 200 characters.

• In the macro processor, the maximum length of a word that is returned by the KSCANX function is 65,534 characters.

The minimum length of the word that is returned by the KSCANX function depends on whether the M modifier is specified. See “Using the SCAN Function with the M Modifier” in SAS Viya Functions and CALL Routines: Reference. See also “Using the SCAN Function without the M Modifier” in SAS Viya Functions and CALL Routines: Reference.

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;
```

Here is the result from the KSCANX function example:

```
substr=sas
```

**KSTRCAT Function**

Concatenates two or more character expressions.

**Categories:** DBCS
CAS

**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 279](#).

**Note:** This function supports the varchar type.

**Tip:** Non-DBCS equivalent function is **CAT** in SAS Viya 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 279 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.

**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 = 漢字文字列の連結</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></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:** Character
  - CAS
- **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 279.
- **Note:** This function supports the varchar type.

**Syntax**

KSTRIP(*string*)

**Required Argument**

*string*

specifies a character constant, variable, or expression.
Details

The KSTRIP function returns the argument with all leading and trailing SBCS blanks removed. If the argument is blank, KSTRIP returns a string with a length of zero.

Assigning the results of 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 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 KSTRIP and STRIP functions return a string with a length of zero, for strings that are blank, and the TRIM function returns a single blank.

The KSTRIP and STRIP functions truncate the output if the receiving variable length is less than the input string. The STRIP function processes only SBCS string truncation. The KSTRIP function processes SBCS and DBCS data.

The KSTRIP and STRIP functions normalize data by removing unnecessary blanks. The KSTRIP function should be used in a DBCS environment. The KSTRIP and STRIP functions behave similarly. The KTRIM and KLEFT functions can also be used for data normalization. However, the KTRIM and KLEFT functions remove all types of DBCS spaces and can cause performance problems, as in this code example:

``` Sas
long_char_variable = KTRIM(KLEFT(long_char_variable));
```

The KSTRIP function resolves these issues. You can modify the code using the KSTRIP function to successfully perform DBCS data normalizations in a DBCS SAS session using this code:

``` Sas
long_char_variable = KSTRIP(long_char_variable);
```

Example

The following example shows how the KSTRIP function deletes leading and trailing blanks, and how the DBCS character is truncated. This example uses the Japanese Shift_JIS encoding.
See Also

Function

• “STRIP Function” in SAS Viya Functions and CALL Routines: Reference

KSUBSTR Function

Extracts a substring from an argument.

Categories: DBCS
CAS

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 279.

Note: This function supports the varchar type.

Tip: See “SUBSTR (right of =) Function” in SAS Viya Functions and CALL Routines: Reference.

Syntax

KSUBSTR(argument, position<n> )

Required Arguments

argument
specifies any SAS character expression.

dposition
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 \textit{argument} after \textit{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 279 for restrictions and more information.

The \texttt{KSUBSTR} function returns a portion of an expression that you specify in \textit{argument}. The portion begins with the character specified by \textit{position} and is the number of characters specified by \( n \).

A variable that is created by \texttt{KSUBSTR} obtains its length from the length of \textit{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>Result=漢字文</td>
</tr>
<tr>
<td>text='漢字文字列の抽出';</td>
<td>Result=字文</td>
</tr>
<tr>
<td>result=ksubstr(text,1,2);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>result=ksubstr(text,3,4);</td>
<td></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=ksubstr(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:
- “\texttt{KSUBSTRB Function}” on page 375
- “\texttt{KUPDATE Function}” on page 381
- “\texttt{KUPDATEB Function}” on page 383
- “\texttt{KUPDATES Function}” on page 384

\texttt{KSUBSTRN Function}
Returns a substring, allowing a result with a length of zero.

\textbf{Category:} DBCS
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8).

Note: This function supports the varchar type.

See: “SUBSTRN Function” in SAS Viya Functions and CALL Routines: Reference and “Internationalization Compatibility for SAS String Functions” on page 279 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>ksubstrn_result=<em>234.56</em></td>
</tr>
<tr>
<td>ksubstrn_result = &quot;&quot;</td>
<td></td>
</tr>
<tr>
<td>put ksubstrn_result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
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 279](#).

**See:** “SUBSTR (right of =) Function” in [SAS Viya Functions and CALL Routines: Reference](#)

---

**Syntax**

```
KSUBSTRB(argument, position<n>)
```

**Required Arguments**

- `argument` specifies any SAS character expression.
- `position` specifies the beginning character position in byte units.
- `n` specifies the length of the substring to extract in byte units.

**Interaction**

If `n` is larger than the length (in byte units) 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 279 for restrictions and more information.

The KSUBSTRB function returns a portion of an expression that you specify in `argument`. The portion begins with the byte unit specified by `position` and is the number of byte units specified by `n`.

A variable that is created by KSUBSTRB obtains its length from the length of `argument`.

---

**Example**

The following example uses Japanese characters.
KTRANSLATE Function

Replaces specific characters in a character expression.

**Categories:**
- DBCS
- CAS

**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 279.

**z/OS specifics:**
This function runs only on z/OS.
Uses the EBCDIC code sequence.

**Note:**
This function supports the varchar type.

**Tip:**
Non-DBCS equivalent function is TRANSLATE in SAS Viya Functions and CALL Routines: Reference.

**Syntax**

\[
\text{KTRANSLATE} (\text{source}, \text{to-1}, \text{from-1} <, \ldots \text{to-n}, \text{from-n}>)
\]

**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 279 for restrictions and more information.

You can use KTRANSLATE to translate a single-byte character expression to a double-byte character expression, or translate a double-byte character expression to a single-byte character expression.

The maximum number of pairs of to and from arguments that KTRANSLATE accepts depends on which operating environment you use to run SAS. There is no functional difference between using several pairs of short arguments, or fewer pairs of longer arguments.

If the KTRANSLATE 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.

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=ktranslate(‘正答’,’回答’,’二課’);</td>
<td>result=正答</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>result=ktranslate(‘abc’,’アイウ’,’abc’);</td>
<td>result=アイウ</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KTRIM Function

Removes trailing DBCS blanks and SO/SI from character expressions.

Categories: DBCS

CAS

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 279.

Note: This function supports the varchar type.
Tip: Non-DBCS equivalent function is “TRIM Function” in SAS Viya 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 279 for restrictions and more information.

KTRIM copies a character argument, removes all trailing blanks, and returns the trimmed argument as a result. 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 12.6 on page 351 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 327
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 279.

### Syntax

\[ \text{KTRUNCATE} \left( \text{argument}, \text{number}, \text{length} \right) \]

### 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 279 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.
**KUPCASE Function**

Converts all single-width English alphabet letters in an argument to uppercase.

**Categories:**
- DBCS
- CAS

**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 279.

**Note:**
This function supports the varchar type.

**Tip:**
See “UPCASE Function” in SAS Viya Functions and CALL Routines: Reference.

**Syntax**

```
KUPCASE(argument)
```

**Required Argument**

`argument`

specifies any SAS character expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 279 for restrictions and more information.

The KUPCASE function copies a character argument, converts all lowercase single-width English alphabet letters to uppercase letters, and returns the altered value as a result.

**Example**

The following example uses Japanese characters.

```
data _null_;  
x1=ktruncate(’漢字のテスト’,6);  
x2=ktruncate(’漢字のテスト’,7);  
x3=ktruncate(’漢字のテスト’,9);  
y1=ktruncate(’漢字のテスト’,5,9);  
y2=ktruncate(’漢字のテスト’,6,6);  
y3=ktruncate(’漢字のテスト’,7,6);  
put x1= / x2= / x3= / y1= / y2= / y3=;  
run;
```
KUPDATE Function

Graphs, deletes, and replaces character value contents.

<table>
<thead>
<tr>
<th>Categories</th>
<th>DBCS</th>
<th>CAS</th>
</tr>
</thead>
</table>

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 279.

Note: This function supports the varchar type.

Syntax

KUPDATE(argument,position,n<, characters-to-replace> )
KUPDATE(argument, position<n> , characters-to-replace)  

Required Arguments

argument
- specifies a character variable.

position
- specifies a numeric expression that is the beginning character position.

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 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 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.

KUPDATE Function

data _null_
  result=kupcase('abcあいうえお');
  put result=;
run;

result=ABCあいうえお
Tip

Enclose a literal string of characters in quotation marks.

Details

See “Internationalization Compatibility for SAS String Functions” on page 279 for restrictions and more information.

The KUPDATE function replaces the value of argument with the expression in characters-to-replace starting at the character that you specify in position.

If the 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.

Examples

Example 1

The following example shows the difference between KUPDATE and KUPDATES.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>str='123456';</td>
<td></td>
</tr>
<tr>
<td>x1=str;  substr(x1,2,3)='';</td>
<td></td>
</tr>
<tr>
<td>y1=kupdate(str,2,3);</td>
<td></td>
</tr>
<tr>
<td>z1=kupdates(str,2,3);</td>
<td></td>
</tr>
<tr>
<td>put x1= / y1= / z1=;</td>
<td></td>
</tr>
<tr>
<td>x2=str;  substr(x2,2,3)='abcd';</td>
<td></td>
</tr>
<tr>
<td>y2=kupdate(str,2,3,'abcd');</td>
<td></td>
</tr>
<tr>
<td>z2=kupdates(str,2,3,'abcd');</td>
<td></td>
</tr>
<tr>
<td>put x2= / y2= / z2=;</td>
<td></td>
</tr>
<tr>
<td>x3=str;  substr(x3,2,3)='ab';</td>
<td></td>
</tr>
<tr>
<td>y3=kupdate(str,2,3,'ab');</td>
<td></td>
</tr>
<tr>
<td>z3=kupdates(str,2,3,'ab');</td>
<td></td>
</tr>
<tr>
<td>put x3= / y3= / z3=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

Example 2

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>x1=kupdate('感じのテスト',1,2,'漢字');</td>
<td>x1=漢字のテスト</td>
</tr>
<tr>
<td>x2=kupdate(x1,1,2,'kanji');</td>
<td>x2=kaのテスト</td>
</tr>
<tr>
<td>x3=kupdate(x1,1,3);</td>
<td>x3=テスト</td>
</tr>
<tr>
<td>x4=kupdate(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>
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 279.

Syntax

\[
\text{KUPDATEB}(\text{argument, position}, n<, \text{characters-to-replace}>)
\]

\[
\text{KUPDATEB}(\text{argument, position}, n> , \text{characters-to-replace})
\]

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 279 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.

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>x1=kupdateb('漢字のテスト','1,9','漢字');</td>
<td>x1=漢字 テスト</td>
</tr>
<tr>
<td>x2=kupdateb(x1,1,9,'kanji');</td>
<td>x2=kanji テスト</td>
</tr>
<tr>
<td>x3=kupdateb(x1,1,9);</td>
<td>x3=テスト</td>
</tr>
<tr>
<td>x4=kupdateb(x1,7,'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>

See Also

Function:
- “KUPDATE Function” on page 381

KUPDATES Function

Inserts, deletes, and replaces character value contents.

**Categories:** DBCS
CAS

**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 279.

**Note:** This function supports the varchar type.

**Syntax**

\[
\text{KUPDATES}(\text{argument},\text{position},n<, \text{characters-to-replace}> )
\]
KUPDATES(argument,position<,n>, 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 *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 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 279 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.

**Examples**

**Example 1**

The following example uses Japanese characters.
Example 2

The following example shows the difference between KUPDATE and KUPDATES.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>str='123456';</td>
<td></td>
</tr>
<tr>
<td>x1=str; substring(x1,2,3)='';</td>
<td></td>
</tr>
<tr>
<td>y1=kupdate(str,2,3);</td>
<td></td>
</tr>
<tr>
<td>z1=kupdates(str,2,3);</td>
<td></td>
</tr>
<tr>
<td>put x1= / y1= / z1=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

x1=1 56
y1=156
z1=1 56

See Also

Functions:
- “KUPDATE Function” on page 381

KVERIFY Function

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

Categories: DBCS
CAS

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 279.

Tip: See “VERIFY Function” in SAS Viya 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 279 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>;</td>
<td>invalid grade value: 可</td>
</tr>
<tr>
<td>input grade :$2. @@;</td>
<td></td>
</tr>
<tr>
<td>check='良否';</td>
<td></td>
</tr>
<tr>
<td>x=kverify(grade,check);</td>
<td></td>
</tr>
<tr>
<td>if x gt 0 then put 'Invalid grade value: ' grade;</td>
<td></td>
</tr>
<tr>
<td>cards:</td>
<td></td>
</tr>
<tr>
<td>良 否 良 否 良 否 可</td>
<td></td>
</tr>
</tbody>
</table>

KVERIFYB Function

Returns the position of the first character 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 279.
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

The KVERIFYB function 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 byte-based value. KVERIFY returns character-based value. When process SBCS (For example, wlat1 encoding) string, they return identical result. But, in a DBCS session (For example, EUC-JP or SHIFT-JIS encoding), most CJK characters are 2-byte width, that makes these functions return different result.

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>check=漢字';</td>
<td>x1=3</td>
</tr>
<tr>
<td>text=漢字の検索';</td>
<td>x2=7</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.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>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 279.</td>
</tr>
</tbody>
</table>
Syntax

\texttt{NLDATE\texttt{(date,descriptor)}}

\textbf{Required Arguments}

\textit{date}

specifies a SAS date value.

\textit{descriptor}

is a variable or expression that specifies how dates and times are formatted in output. The following descriptors are case sensitive:

\begin{itemize}
  \item \texttt{#}
    \begin{itemize}
      \item removes the leading zero from the result.
    \end{itemize}
  \item \texttt{%%}
    \begin{itemize}
      \item specifies the \% character.
    \end{itemize}
  \item \texttt{%%a}
    \begin{itemize}
      \item specifies the short-weekday descriptor. The range for the day descriptor is Mon–Sun.
    \end{itemize}
  \item \texttt{%%A}
    \begin{itemize}
      \item specifies the long-weekday descriptor. The range for the long-weekday descriptor is Monday–Sunday.
    \end{itemize}
  \item \texttt{%%b}
    \begin{itemize}
      \item specifies the short-month descriptor. The range for the short-month descriptor is Jan–Dec.
    \end{itemize}
  \item \texttt{%%B}
    \begin{itemize}
      \item specifies the long-month descriptor. The range for the long-month descriptor is January–December.
    \end{itemize}
  \item \texttt{%%C}
    \begin{itemize}
      \item specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.
    \end{itemize}
  \item \texttt{%%d}
    \begin{itemize}
      \item specifies the day descriptor and uses 0 padding. The range for the day modifier is 01–31.
    \end{itemize}
  \item \texttt{%%e}
    \begin{itemize}
      \item specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.
    \end{itemize}
  \item \texttt{%%F}
    \begin{itemize}
      \item specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.
    \end{itemize}
  \item \texttt{%%j}
    \begin{itemize}
      \item 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.
    \end{itemize}
  \item \texttt{%%m}
    \begin{itemize}
      \item specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.
    \end{itemize}
\end{itemize}
%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>
</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=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>logfile=nldate('24Feb2003'd,'%B-%d.log');</td>
<td>Februar-24.log</td>
</tr>
<tr>
<td>put logfile;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_unitedstates;</td>
<td></td>
</tr>
<tr>
<td>weekname=nldate('24Feb2003'd,'%A');</td>
<td>Monday</td>
</tr>
<tr>
<td>put weekname;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>weekname=nldate('24Feb2003'd,'%A');</td>
<td>Montag</td>
</tr>
<tr>
<td>put weekname;</td>
<td></td>
</tr>
</tbody>
</table>

## See Also

Format:
- “NLDATE Format” on page 106

---

**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 279.

### Syntax

```
NLDATM(datetime,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.

d specifies the day of the year descriptor as a decimal number and uses a leading zero. The range for the day descriptor is 1–366.

%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 are 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 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>---+---+---1---+</td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
options locale=English; | 12PM
```
time_ampm=nldatm('24Feb2003:12:39:43' dt, '%I%p');
put time_ampm;
```

options locale=German; | 12nachm
```
time_ampm=nldatm('24Feb2003:12:39:43' dt, '%I%p');
put time AMPM;
```

See Also

Format:

- “NLDATM Format” on page 127

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 279.

**Syntax**

\[ \text{NLTIME}(\text{time|datetime,descriptor,startpos}) \]

**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.
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.

\textit{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>12 PM</td>
</tr>
<tr>
<td>options locale=German; time_ampm=nltime('12:39:43't,'%i%p'); put time_ampm;</td>
<td>12 nachm</td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “NLTIME Format” on page 242

\textbf{SASMSG Function}

Specifies a message from a data set. The returned message is based on the current locale and a specified key.

\textbf{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 279.

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:

```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 the order:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg = sasmsg</td>
<td>msg= My cat. Your dog.</td>
</tr>
<tr>
<td>(&quot;nls.mymsg&quot;,&quot;IN_CD_LOG&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;noquote&quot;,&quot;cat&quot;,&quot;dog&quot;);</td>
<td></td>
</tr>
<tr>
<td>IN_CD_LOGINFO = My %#1s. Your %#2s</td>
<td>msg= My dog. Your cat.</td>
</tr>
</tbody>
</table>

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;
### 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 279.

#### Syntax

```
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 (ll_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>msg = sasmsgl(&quot;nls.mymsg&quot;,&quot;IN_CD_Log&quot;,&quot;en_US&quot;,&quot;N&quot;,&quot;cat&quot;,&quot;dog&quot;);</td>
<td>msg= My cat. Your dog.</td>
</tr>
<tr>
<td>IN_CD_LOGINFO = My %#1s. Your %#2s</td>
<td>IN_CD_LOGINFO = My %#2s. Your %#1s</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(&quot;nls.mymsg&quot;, &quot;IN_APW_SAVE_OK&quot;, &quot;en_US&quot;, &quot;n&quot;);</td>
<td>The Access Control key was successfully saved.</td>
</tr>
<tr>
<td>sasmsgl(&quot;nls.mymsg&quot;, &quot;IN_APW_SAVE_OK&quot;, &quot;es_ES&quot;, &quot;n&quot;);</td>
<td>La clave de control de acceso se ha guardado.</td>
</tr>
<tr>
<td>sasmsgl(&quot;nls.mymsg&quot;, &quot;IN_APW_SAVE_OK&quot;, &quot;fr&quot;, &quot;n&quot;);</td>
<td>La clé de contrôle d'accès a bien été enregistrée.</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></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 279.

Syntax

sortKey(string, <locale, strength, case_order, numeric_order,> )

Required Arguments

string
character expression

locale
specifies the locale name in the form of a POSIX name (ja_JP). See Table 17.1 on page 629 for a list of locale names and POSIX values.

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;âs&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>
</tbody>
</table>
**TERTIARY or T**  Upper and lowercase differences in characters are distinguished at the tertiary level (for example, "ao" < "Ao" < "aò"). 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.

**QUATERNARY or Q**  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.

**IDENTICAL or I**  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**

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 <strong>PHONEBOOK</strong> only with the German 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 processes 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>
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.

---

**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 279.

**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
- LC_ALL

**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_SANS_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_CURLSIVE</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_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>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>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>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>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_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>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>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>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>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>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>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>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>
</tbody>
</table>
Locale Element Key | Max Length | Type | Category
--- | --- | --- | ---
THOUSANDS_SEP | 1 | 0 | LC_NUMERIC
GROUPING | 3 | 1 | LC_NUMERIC
POSITIVE_SIGN | 8 | 0 | LC_NUMERIC
NEGATIVE_SIGN | 8 | 0 | LC_NUMERIC
P_CS_PRECEDES | 3 | 1 | LC_NUMERIC
P_SEP_BY_SPACE | 3 | 1 | LC_NUMERIC
N_CS_PRECEDES | 3 | 1 | LC_NUMERIC
P_SEP_BY_SPACE | 3 | 1 | LC_NUMERIC
N_CS_PRECEDES | 3 | 1 | LC_NUMERIC
N_SEP_BY_SPACE | 3 | 1 | LC_NUMERIC
P_SIGN_POSN | 3 | 1 | LC_NUMERIC
N_SIGN_POSN | 3 | 1 | LC_NUMERIC
HEIGHT | 3 | 1 | 
WIDTH | 3 | 1 | 

Examples

Example 1
In the following locale example, the SETLOCALE function specifies the locale Japanese (ja_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>
</table>
data _null_; x=setlocale('ja_JP'); put x=; run; | x=English_UnitedStates |

Example 2
In the following example, the SETLOCALE function returns the locale name where the element values are being changed:
Statements | Results
---|---
data _null_; | x=Japanese_Japan
x = setlocale("LC_MONETARY", 'zh_CN'); | put x=; run; | x=¥%

**Example 3**
In the following example, the SETLOCALE function changes the value of the specified key, DATE_YEAR_FORMAT:

Statements | Results
---|---
data _null_; | x=¥%
x=setlocale('DATE_YEAR_FORMAT', '¥%Y'); | put x=; run; | x=¥%

---

**TZONEID Function**

Returns the current time zone ID.

**Category:** Date and Time

**Alias:** TZID

**Restrictions:**
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 279.

This function is not valid in the CAS server.

**Syntax**

TZONEID<time-zone-id>

**Optional Argument**

`time-zone-id`

specifies a region or 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 time zone 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>options timezone=jst; data <em>null</em>; tzid=tzoneid(); put tzid=; run;</td>
<td>tmid=ASIA/TOKYO</td>
</tr>
<tr>
<td>options timezone=''; data <em>null</em>; tzid=tzoneid(); put tzid=; run;</td>
<td>tmid=</td>
</tr>
<tr>
<td>options timezone='xxx-12'; /* user defined timezone */ data <em>null</em>; tzid=tzoneid(); put tzid=; run;</td>
<td>tmid=</td>
</tr>
<tr>
<td>data null; name_valid=tzoneid('asia/tokyo'); name_invalid=tzoneid('Milky Way'); put name_valid =; put name_invalid=; run;</td>
<td>name_valid=ASIA/TOKYO name_invalid=</td>
</tr>
</tbody>
</table>

TZONENAME Function

Returns the current standard or daylight savings time, time zone name.

Category: Date and Time

Restrictions: 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 279.

This function is not valid in the CAS server.

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 2, “Time Zone IDs and Time Zone Names,” on page 663.

datetime

specifies a SAS datetime value.

Details

The TZONENAME function returns a blank value if the TIMEZONE= option is 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 TZID 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 timezone ID.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| options tz='';
data _null_;
tzname=tzonename();
put tzname =; run; | tzone= |
| options tz='jst';
data _null_;
tzname=tzonename();
put tzname =; run; | tzone=JST |
| options tz='xxx-12';
data _null_;
tzname=tzonename();
put tzname =; run; | tzone=XXX |
| options tz='American/Chicago';
data_null;
tzname=tzonename('01SEP2014:01:01:01'dt);
put tzname =; run; | tzone=CDT |
**TZONEOFF Function**

Returns the user time zone offset.

**Category:** Date and Time

**Restrictions:** 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 279](#).

This function is not valid in the CAS server.

**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 2, “Time Zone IDs and Time Zone Names,” on page 663.

- **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 argument returns the time zone offset for the specified time zone name. If the time zone name is not valid for the current locale, you receive an error. 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.
Statements | Results
---|---
\```sas
option TIMEZONE='AUSTRALIA/MELBOURNE';
%PUT %SYSFUNC(TZONEOFF());
```
39600

\```sas
option TIMEZONE='AUSTRALIA/MELBOURNE';
%PUT %SYSFUNC(TZONEOFF('EUROPE/ROME'));
```
3600

\```sas
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 ;
```
offset1=10800
offset2=14400

\```sas
option 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 ;
```
offset1=10800
offset2=14400

TZONES2U Function

Converts a SAS date time value to a UTC date time value.

**Category:** Date and Time

**Restrictions:** 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 279.

This function is not valid in the CAS server.

**Syntax**

`TZONES2U <datetime, time-zone-id>`

**Optional Arguments**

- `datetime`
  
  specifies a SAS datetime value.

- `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 time zone name and daylight savings time rules.

**See** For a list of time zone IDs, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 663
Details

The TZONE2U() function returns UTC-based time for the specified TIMEZONE. The TZONE2U(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.

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> ;</td>
<td></td>
</tr>
<tr>
<td>dt='05Nov2012:08:17:52'dt ;</td>
<td>utc1=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>utc1 = tzones2u(dt) ;</td>
<td></td>
</tr>
<tr>
<td>utc2 = tzones2u(dt,'ASIA/TOKYO') ;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>utc3 = tzones2u(dt,'JST') ;</td>
<td>utc2=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>put dt = /utc1= is8601dz. //;</td>
<td></td>
</tr>
<tr>
<td>put dt = /utc2= is8601dz. //;</td>
<td></td>
</tr>
<tr>
<td>put dt = /utc3= is8601dz. //;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>run ;</td>
<td>utc3=2012-11-04T23:17:52+00:00</td>
</tr>
</tbody>
</table>

TZONEDSTNAME Function

Returns a daylight savings time name.

Category: Date and Time

Restrictions: 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 279.

This function is not valid in the CAS server.

Syntax

TZONEDSTNAME()
TZONEDSTNAME<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. If the daylight savings time is not specified, 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';</td>
<td>dstname=CDT</td>
</tr>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>dstname=tzonedstname();</td>
<td></td>
</tr>
<tr>
<td>put dstname;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

TZONEDSTOFF Function

Returns the time zone offset value for the specified daylight savings time.

**Category:** Date and Time

**Restrictions:** 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 279.

This function is not valid in the CAS server.

**Syntax**

TZONEDSTOFF()

TZONEDSTOFF<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. If the daylight savings time is not specified, then the function returns a blank name.

**Example**

The following example specifies the America/Chicago time zone.
### TZONESTTNAME Function

Returns a standard time zone name.

#### Category:
Date and Time

#### Restrictions:
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 279.

This function is not valid in the CAS server.

#### Syntax

\[ \text{TZONESTTNAME()} \]

\[ \text{TZONESTTNAME<time-zone id>} \]

#### 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options timezone='Asia/Osaka';</code></td>
<td>JST</td>
</tr>
<tr>
<td><code>data null;</code></td>
<td></td>
</tr>
<tr>
<td><code>name=tzonesttname();</code></td>
<td></td>
</tr>
<tr>
<td><code>put name;</code></td>
<td></td>
</tr>
<tr>
<td><code>run;</code></td>
<td></td>
</tr>
</tbody>
</table>
TZONESTTOFF Function

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

**Category:** Date and Time

**Restrictions:** 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 279.

This function is not valid in the CAS server.

**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 TZID 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>
<tbody>
<tr>
<td>options timezone='Asia/Osaka'; data null; name=tzonesttoff(); put name; run;</td>
<td>32400</td>
</tr>
</tbody>
</table>

TZONEU2S Function

Converts a UTC date time value to a SAS date time value.

**Category:** Date and Time

**Restrictions:** 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 279.
This function is not valid in the CAS server.

**Syntax**

```
TZONETWOS <UTC date time value, time-zone id>
```

**Optional Arguments**

- **UTC date time value**: specifies a Coordinated Universal Time (UTC) datetime value
- **time-zone id**: specifies a region or 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. If the time zone name is not valid for the current locale, you receive an error.

**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';</td>
<td>sdt1=01SEP12:20:34:56</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>sdt2=02SEP12:03:34:56</td>
</tr>
<tr>
<td>utc_date = '2012-09-02T02:34:56+00:00';</td>
<td>sdt3=02SEP12:03:34:56</td>
</tr>
<tr>
<td>udt = input(utc_date,is8601dz.);</td>
<td></td>
</tr>
<tr>
<td>sdt1 = tzoneu2s(udt);</td>
<td></td>
</tr>
<tr>
<td>sdt2 = tzoneu2s(udt,'EUROPE/AMSTERDAM');</td>
<td></td>
</tr>
<tr>
<td>sdt3 = tzoneu2s(udt,'CET');</td>
<td></td>
</tr>
<tr>
<td>put sdt1= datetime. / sdt2= datetime. / sdt3= datetime.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**TRANTAB Function**

Transcodes data by using the specified translation table.

**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 279](#).
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');</td>
<td>TESTING</td>
</tr>
<tr>
<td>put teststrg;</td>
<td></td>
</tr>
</tbody>
</table>

UNICODE Function

Converts Unicode characters to the current SAS session encoding.

Category: Character

Restrictions: 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 279.

UTF-8 is the only SAS session encoding supported by SAS Viya.
**Syntax**

\[ \text{STR=} \text{UNICODE}(\text{<instr>}, (\text{<Unicode type>}) ) \]

**Required Arguments**

- \( \text{str} \)
  - Data string that has been converted to the current SAS session encoding.
- \( \text{instr} \)
  - Input data string.
- \( \text{Unicode type} \)
  - Unicode character formats
    - ESC: Unicode Escape (for example, \( \text{\u0042} \)). ESC is the default format.
    - NCR: Numeric Character Representation (for example, \( \&\#22823 \) or \( \&\#177; \)).
    - PAREN: Unicode Parenthesis Escape (for example, \( \text{<u0061>} \)).
    - 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:

```plaintext
/* Run this program using any CJK locale */
data _null_;  /* default */
str1=unicode("\u0041\u0042\u0043");put str1=;
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 */
```
UNICODEC Function

Converts characters in the current SAS session encoding to Unicode characters.

**Category:** Character

**Restrictions:**
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 279.

UTF-8 is the only SAS session encoding supported by SAS Viya.

**Syntax**

```sas
STR=UNICODEC(<instr> ,<Unicode type> ))
```

**Required Arguments**

- `str`  
  data string that has been converted to Unicode encoding.

- `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 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.

Example
The following example demonstrates the functionality of the UNICODEC function:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>str1=unicodc(&quot;ABC&quot;, 'utf8');</td>
<td>str1=414243202020</td>
</tr>
<tr>
<td>put str1= $hex12.;</td>
<td></td>
</tr>
<tr>
<td>str2=unicodc(&quot;ABCé&quot;, 'utf8');</td>
<td>str2=414243C3A920</td>
</tr>
<tr>
<td>put str2= $hex12.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

UNICODELEN 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 279.

Syntax
UNICODELEN()

Details
The UNICODELEN function specifies the length of the character unit for the UNICODE data.
Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1=unicodelen(&quot;abcña&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;大&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 425

UNICODEWIDTH Function

Specifies the length of a display unit for the Unicode data.

**Category:** Character

**Restriction:** This function is assigned an l18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 279.

Syntax

UNICODEWIDTH()

Details

The UNICODEWIDTH function specifies the length of a display unit for the Unicode data. The display unit displays the width of a character when the character is displayed with fixed width font.

The characters in CJK Symbols, CJK Unified Ideographs, Full width Alphabets and Punctuation, Full width currency symbols, CJK Unified Ideograph Extension B-D and CJK Compatibility Ideographs Supplement have the value of a display unit 2. Other characters are display unit 1.

Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>


UNICODEWIDTH Function
len1=unicodewidth("abc.selectAll"); len1=5
len2=unicodewidth("\u0041\u0042\u0043\u5927","esc"); len2=5
len3=unicodewidth("&#22823; ","ncr"); len3=2
len4=unicodewidth("<u0061><u0062>","paren"); len4=2

See Also

Functions:
• “UNICODELEN Function” on page 424

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_;
dsids=open('work.a','i');
nobs=attrn(dsids,"nobs");
nvars=attrn(dsids,"nvars");
do i=1 to nobs;
xrc=fetch(dsids,1);
do j=1 to nvars;
   transcode = vartranscode(dsids,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 Viya Functions and CALL Routines: Reference
- “OPEN Function” in SAS Viya Functions and CALL Routines: Reference
- “VARNUM Function” in SAS Viya Functions and CALL Routines: Reference
- “VTRANSCODE Function” on page 427
- “VTRANSCODEX Function” on page 428

VTRANSCODE Function

Returns a value that indicates whether transcoding is enabled for the specified character variable.

Categories: Variable Information
CAS

Note: This function supports the varchar type.

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 Viya Functions and CALL Routines: Reference.

Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
</table>
| attrib x transcode = yes; 
attrib y transcode = no; 
rcl = vtranscode(y); 
put rcl=; | rcl=0 |

See Also

Functions:
• “VTRANSCODEX Function” on page 428

Statements:
• ATTRIB in

VTRANSCODEX Function
Returns a value that indicates whether transcoding is enabled for the specified argument.

Categories: Variable Information
CAS

Note: This function supports the varchar type.
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 Viya 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></td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td></td>
</tr>
<tr>
<td>rcl = vtranscodex('y');</td>
<td>rcl=0</td>
</tr>
<tr>
<td>put rcl=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:

• “VTRANSCODE Function” on page 427

Statements:

• ATTRIB
Part 6

Informats for NLS

Chapter 13

Dictionary of Informats for NLS
Chapter 13
Dictionary of Informats for NLS

Informats by Category

435

Dictionary . ................................................................. 442
$CP$TDW Informat ......................................................... 442
$CP$TWD Informat ......................................................... 443
EURO Informat ............................................................. 444
EUROX Informat ........................................................... 445
MINGUO Informat .......................................................... 447
NENGO Informat ............................................................ 449
NLDATE Informat ........................................................... 450
NLDATEW Informat ......................................................... 451
NLDATM Informat ........................................................... 452
NLDATMAP Informat ....................................................... 453
NLDATMW Informat ........................................................ 454
NLMNIAED Informat ........................................................ 455
NLMNIAUD Informat ........................................................ 456
NLMNIBGN Informat ......................................................... 457
NLMNIBRL Informat ......................................................... 458
NLMNICAD Informat ........................................................ 459
NLMNICHF Informat ......................................................... 460
NLMNICNY Informat ........................................................ 461
NLMNICZK Informat ........................................................ 462
NLMNIDKK Informat ........................................................ 463
NLMNIEEK Informat ........................................................ 464
NLMNIEGP Informat ........................................................ 465
NLMNIEUR Informat ........................................................ 466
NLMNIGBP Informat ........................................................ 467
NLMNIHKD Informat ........................................................ 468
NLMNIHRK Informat ........................................................ 469
NLMNIHUF Informat ........................................................ 470
NLMNIIDR Informat ........................................................ 471
NLMNIILS Informat ........................................................ 472
NLMNIINR Informat ........................................................ 473
NLMNIJPY Informat ........................................................ 474
NLMNIKRW Informat ........................................................ 475
NLMNILTL Informat ........................................................ 476
NLMNILVL Informat ........................................................ 477
NLMNIMOP Informat ........................................................ 478
NLMNIMXN Informat ........................................................ 479
NLMNIMYR Informat ........................................................ 480
NLMNINOK Informat ........................................................ 481
NLMNINZD Informat ........................................................ 482
NLNIPLN Informat ..................................................... 483
NLNIRUB Informat ..................................................... 484
NLNISEK Informat ..................................................... 485
NLNISGD Informat ..................................................... 486
NLNITHB Informat ..................................................... 487
NLNITRY Informat ..................................................... 488
NLNITWD Informat ..................................................... 489
NLNIUSD Informat ..................................................... 490
NLNIZAR Informat ..................................................... 491
NLNLAEED Informat .................................................. 492
NLNLAUD Informat .................................................... 493
NLNLBGN Informat .................................................... 494
NLNLBRL Informat .................................................... 495
NLNLBRCAD Informat ................................................. 496
NLNLCHF Informat .................................................... 497
NLNLCHF Informat .................................................... 498
NLNLTZK Informat .................................................... 499
NLNLNOK Informat ................................................... 500
NLNLNOK Informat ................................................... 501
NLNLNOK Informat ................................................... 502
NLNLNOK Informat ................................................... 503
NLNLNOK Informat ................................................... 504
NLNLNOK Informat ................................................... 505
NLNLNOK Informat ................................................... 506
NLNLNOK Informat ................................................... 507
NLNLNOK Informat ................................................... 508
NLNLNOK Informat ................................................... 509
NLNLNOK Informat ................................................... 510
NLNLNOK Informat ................................................... 511
NLNLNOK Informat ................................................... 512
NLNLNOK Informat ................................................... 513
NLNLNOK Informat ................................................... 514
NLNLNOK Informat ................................................... 515
NLNLNOK Informat ................................................... 516
NLNLNOK Informat ................................................... 517
NLNLNOK Informat ................................................... 518
NLNLNOK Informat ................................................... 519
NLNLNOK Informat ................................................... 520
NLNLNOK Informat ................................................... 521
NLNLNOK Informat ................................................... 522
NLNLNOK Informat ................................................... 523
NLNLNOK Informat ................................................... 524
NLNLNOK Informat ................................................... 525
NLNLNOK Informat ................................................... 526
NLNLNOK Informat ................................................... 527
NLNLNOK Informat ................................................... 528
NLNLNOK Informat ................................................... 529
NLNLNOK Informat ................................................... 530
NLNLNOK Informat ................................................... 531
NLNLNOK Informat ................................................... 532
NLNLNOK Informat ................................................... 533
NLNLNOK Informat ................................................... 534
NLNLNOK Informat ................................................... 535
NLNLNOK Informat ................................................... 536
NLNLNOK Informat ................................................... 537
NLNLNOK Informat ................................................... 537
NLNLNOK Informat ................................................... 538
NLNLNOK Informat ................................................... 538
NLNLNOK Informat ................................................... 539
There are six categories of SAS informats that support NLS:

<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>Character</td>
<td>Instructs SAS to read character data values into character variables.</td>
</tr>
<tr>
<td>DBCS</td>
<td>Instructs SAS to manage various Asian languages.</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>
</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>Character</td>
<td>$UCS2B Informat (p. 538)</td>
<td>Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$UCS2BE Informat (p. 539)</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>
<td></td>
</tr>
<tr>
<td>$UCS2L Informat (p. 540)</td>
<td>Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS2LE Informat (p. 542)</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>
<td></td>
</tr>
<tr>
<td>$UCS2X Informat (p. 543)</td>
<td>Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS2XE Informat (p. 544)</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>
<td></td>
</tr>
<tr>
<td>$UCS4B Informat (p. 545)</td>
<td>Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4L Informat (p. 546)</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>
<td></td>
</tr>
<tr>
<td>$UCS4X Informat (p. 547)</td>
<td>Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4XE Informat (p. 548)</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>
<td></td>
</tr>
<tr>
<td>$UESC Informat (p. 549)</td>
<td>Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UESCE Informat (p. 550)</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>
<td></td>
</tr>
<tr>
<td>$UNCR Informat (p. 551)</td>
<td>Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UNCRE Informat (p. 552)</td>
<td>Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.</td>
<td></td>
</tr>
<tr>
<td>$UPAREN Informat (p. 553)</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>
<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>SUPARENE Informat (p. 555)</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></td>
<td>SUPARENP Informat (p. 556)</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></td>
<td>SUTF8X Informat (p. 557)</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></td>
<td></td>
<td>Date and Time</td>
</tr>
<tr>
<td></td>
<td>MINGUO Informat (p. 447)</td>
<td>Reads dates in Taiwanese format.</td>
</tr>
<tr>
<td></td>
<td>NENGO Informat (p. 449)</td>
<td>Reads Japanese date values in the form eyymmdd.</td>
</tr>
<tr>
<td></td>
<td>NLDATE Informat (p. 450)</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>NLDATEW Informat (p. 451)</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>NLDATM Informat (p. 452)</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. 453)</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></td>
<td>NLDATMW Informat (p. 454)</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></td>
<td>NLTIMAP Informat (p. 537)</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>NLTIME Informat (p. 537)</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></td>
<td></td>
<td>Hebrew Text Handling</td>
</tr>
<tr>
<td></td>
<td>$CPTDW Informat (p. 442)</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></td>
<td>$CPTWD Informat (p. 443)</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></td>
<td></td>
<td>Numeric</td>
</tr>
<tr>
<td></td>
<td>EURO Informat (p. 444)</td>
<td>Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.</td>
</tr>
<tr>
<td></td>
<td>EUROX Informat (p. 445)</td>
<td>Reads numeric values and removes embedded characters in European currency.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLMNIAED Informat (p. 455)</td>
<td>NLMNIAED Informat (p. 455)</td>
<td>Reads the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td>NLMNIAUD Informat (p. 456)</td>
<td>NLMNIAUD Informat (p. 456)</td>
<td>Reads the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td>NLMNIBGN Informat (p. 457)</td>
<td>NLMNIBGN Informat (p. 457)</td>
<td>Reads the monetary format of the international expression for Bulgaria.</td>
</tr>
<tr>
<td>NLMNIBRL Informat (p. 458)</td>
<td>NLMNIBRL Informat (p. 458)</td>
<td>Reads the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td>NLMNICAD Informat (p. 459)</td>
<td>NLMNICAD Informat (p. 459)</td>
<td>Reads the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td>NLMNICCHF Informat (p. 460)</td>
<td>NLMNICCHF Informat (p. 460)</td>
<td>Reads the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td>NLMNICNY Informat (p. 461)</td>
<td>NLMNICNY Informat (p. 461)</td>
<td>Reads the monetary format of the international expression for China.</td>
</tr>
<tr>
<td>NLMNICZK Informat (p. 462)</td>
<td>NLMNICZK Informat (p. 462)</td>
<td>Reads the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td>NLMNIDKK Informat (p. 463)</td>
<td>NLMNIDKK Informat (p. 463)</td>
<td>Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td>NLMNIEEK Informat (p. 464)</td>
<td>NLMNIEEK Informat (p. 464)</td>
<td>Reads the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td>NLMNIEGP Informat (p. 465)</td>
<td>NLMNIEGP Informat (p. 465)</td>
<td>Reads the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td>NLMNIEUR Informat (p. 466)</td>
<td>NLMNIEUR Informat (p. 466)</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>NLMNIGBP Informat (p. 467)</td>
<td>NLMNIGBP Informat (p. 467)</td>
<td>Reads the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td>NLMNIHKD Informat (p. 468)</td>
<td>NLMNIHKD Informat (p. 468)</td>
<td>Reads the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td>NLMNIIHRK Informat (p. 469)</td>
<td>NLMNIIHRK Informat (p. 469)</td>
<td>Reads the monetary format of the international expression for Croatia.</td>
</tr>
<tr>
<td>NLMNIIHUF Informat (p. 470)</td>
<td>NLMNIIHUF Informat (p. 470)</td>
<td>Reads the monetary format of the international expression for Hungary.</td>
</tr>
<tr>
<td>NLMNIIIDR Informat (p. 471)</td>
<td>NLMNIIIDR Informat (p. 471)</td>
<td>Reads the monetary format of the international expression for Indonesia.</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>NLMNIILS Informat (p. 472)</td>
<td>Reads the monetary format of the international expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNIINR Informat (p. 473)</td>
<td>Reads the monetary format of the international expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNIJPY Informat (p. 474)</td>
<td>Reads the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>NLMNIKRW Informat (p. 475)</td>
<td>Reads the monetary format of the international expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNILTL Informat (p. 476)</td>
<td>Reads the monetary format of the international expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNILVL Informat (p. 477)</td>
<td>Reads the monetary format of the international expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIMOP Informat (p. 478)</td>
<td>Reads the monetary format of the international expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>NLMNIMXN Informat (p. 479)</td>
<td>Reads the monetary format of the international expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>NLMNIMYR Informat (p. 480)</td>
<td>Reads the monetary format of the international expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>NLMNINOK Informat (p. 481)</td>
<td>Reads the monetary format of the international expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMNINZD Informat (p. 482)</td>
<td>Reads the monetary format of the international expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNIIPLN Informat (p. 483)</td>
<td>Reads the monetary format of the international expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNIRUB Informat (p. 484)</td>
<td>Reads the monetary format of the international expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>NLMNISEK Informat (p. 485)</td>
<td>Reads the monetary format of the international expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>NLMNISGD Informat (p. 486)</td>
<td>Reads the monetary format of the international expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>NLMNITHB Informat (p. 487)</td>
<td>Reads the monetary format of the international expression for Thailand.</td>
</tr>
<tr>
<td></td>
<td>NLMNITRY Informat (p. 488)</td>
<td>Reads the monetary format of the international expression for Turkey.</td>
</tr>
<tr>
<td></td>
<td>NLMNITWD Informat (p. 489)</td>
<td>Reads the monetary format of the international expression for Taiwan.</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>NLMNUSD Informat (p. 490)</td>
<td>Reads the monetary format of the international expression for Puerto Rico and the United States.</td>
</tr>
<tr>
<td></td>
<td>NLMNIZAR Informat (p. 491)</td>
<td>Reads the monetary format of the international expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>NLMNLAED Informat (p. 492)</td>
<td>Reads the monetary format of the local expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>NLMNLAUD Informat (p. 493)</td>
<td>Reads the monetary format of the local expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBGN Informat (p. 494)</td>
<td>Reads the monetary format of the local expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBRL Informat (p. 495)</td>
<td>Reads the monetary format of the local expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCAD Informat (p. 496)</td>
<td>Reads the monetary format of the local expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNICHF Informat (p. 497)</td>
<td>Reads the monetary format of the local expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCHF Informat (p. 498)</td>
<td>Reads the monetary format of the local expression for China.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCHF Informat (p. 499)</td>
<td>Reads the monetary format of the local expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>NLMNLDKK Informat (p. 500)</td>
<td>Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEEK Informat (p. 501)</td>
<td>Reads the monetary format of the local expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEG Informat (p. 502)</td>
<td>Reads the monetary format of the local expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEUR Informat (p. 503)</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>NLMNGBP Informat (p. 504)</td>
<td>Reads the monetary format of the local expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHKD Informat (p. 505)</td>
<td>Reads the monetary format of the local expression for Hong Kong.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHRY Informat (p. 506)</td>
<td>Reads the monetary format of the local expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td>NLMNHF Informat (p. 507)</td>
<td>Reads the monetary format of the local expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>NLMNLIDR Informat (p. 508)</td>
<td>Reads the monetary format of the local expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLS Informat (p. 509)</td>
<td>Reads the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNLINR Informat (p. 510)</td>
<td>Reads the monetary format of the local expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNLJ Informat (p. 511)</td>
<td>Reads the monetary format of the local expression for Japan.</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>NLMNLIKRW Informat (p. 512)</td>
<td>Reads the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLTL Informat (p. 513)</td>
<td>Reads the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLVL Informat (p. 514)</td>
<td>Reads the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMOP Informat (p. 515)</td>
<td>Reads the monetary format of the local expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMXN Informat (p. 516)</td>
<td>Reads the monetary format of the local expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMYR Informat (p. 517)</td>
<td>Reads the monetary format of the local expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNOK Informat (p. 518)</td>
<td>Reads the monetary format of the local expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNZD Informat (p. 519)</td>
<td>Reads the monetary format of the local expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNLPLN Informat (p. 520)</td>
<td>Reads the monetary format of the local expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLRUB Informat (p. 521)</td>
<td>Reads the monetary format of the local expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>NLMNILSEK Informat (p. 522)</td>
<td>Reads the monetary format of the local expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>NLMNLSGD Informat (p. 523)</td>
<td>Reads the monetary format of the local expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTHB Informat (p. 524)</td>
<td>Reads the monetary format of the local expression for Thailand.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTRY Informat (p. 525)</td>
<td>Reads the monetary format of the local expression for Turkey.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTWD Informat (p. 526)</td>
<td>Reads the monetary format of the local expression for Taiwan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLUSD Informat (p. 527)</td>
<td>Reads the monetary format of the local expression for Puerto Rico, and the United States.</td>
</tr>
<tr>
<td></td>
<td>NLMNLZAR Informat (p. 528)</td>
<td>Reads the monetary format of the local expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>NLMNY Informat (p. 529)</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>NLMNYI Informat (p. 530)</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>NLMNUM Informat (p. 531)</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>NLMNUMI Informat (p. 533)</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>NLPCT Informat (p. 534)</td>
<td>Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
</tr>
</tbody>
</table>
**Dictionary**

### $\text{CPTDW Informat}$

Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding.

**Category:** Hebrew Text Handling

**Syntax**

$\text{CPTDW}_w$

**Syntax Description**

$w$

specifies the width of the input field.

- **Default:** 200
- **Range:** 1–32767

**Comparisons**

The $\text{CPTDW}_w$. informat performs processing that is opposite of the $\text{CPTWD}_w$. 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>$x=\text{input}'808182'$,$\text{CPTDW}_6$.; put $x;$</td>
<td>128</td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**
$CPTWD 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

**Syntax**

$CPTWD w.

**Syntax Description**

w

specifies the width of the input field.

<table>
<thead>
<tr>
<th>Default</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>1–32767</td>
</tr>
</tbody>
</table>

**Comparisons**

The $CPTWD w. informat performs processing that is opposite of the $CPTDW w. informat.

**Example**

The following example uses the input value of ישר.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-------</td>
</tr>
<tr>
<td>x=input ('ישר',$cptwd6.);</td>
<td>$[]</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

- “$CPTWD Format” on page 89
- “$CPTDW Format” on page 88
Informat:
• “$CPTDW Informat” on page 442

EURO Informat
Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.

Category: Numeric

Syntax

$$EURO_{w.d}$$

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 $EURO_{w.d}$ 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 $EURO_{w.d}$ informat converts an open parenthesis at the beginning of a field to a minus sign.

Comparisons

• The $EURO_{w.d}$ informat is similar to the $EUROX_{w.d}$ informat, but $EUROX_{w.d}$ 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 $EURO_{w.d}$ and the $EUROX_{w.d}$ 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_;
```sas
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>1234.56</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**
- “EURO Format” on page 96
- “EUROX Format” on page 99

**Informat:**
- “EUROX Informat” on page 445

---

**EUROX Informat**

Reads numeric values and removes embedded characters in European currency.

**Category:** Numeric
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;  
E1  
E1.23  
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>
<tbody>
<tr>
<td>E1</td>
<td>input x eurox10.; put x;</td>
<td>1</td>
</tr>
<tr>
<td>E1.23</td>
<td>input x eurox10.; put x;</td>
<td>123</td>
</tr>
<tr>
<td>1.23</td>
<td>input x eurox10.; put x;</td>
<td>123</td>
</tr>
<tr>
<td>1,234.56</td>
<td>input x eurox10.; put x;</td>
<td>1.23456</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “EURO Format” on page 96
- “EUROX Format” on page 99

Informat:
- “EURO Informat” on page 444

MINGUO Informat

Reads dates in Taiwanese format.

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 `YYMMDw.` 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.

```sas
input date minguo10.;
put date date9.;
data _null_;
  input date minguo10.;
  put date date9.;
datalines;
49/01/01
891215
03-01-01
;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>01JAN1960</td>
<td>01JAN1960</td>
</tr>
<tr>
<td>15DEC2000</td>
<td>15DEC2000</td>
</tr>
<tr>
<td>01JAN2014</td>
<td>01JAN2014</td>
</tr>
</tbody>
</table>

See Also

Format:

- “MINGUO Format” on page 101

Informat:

- “YYMMDD Informat” in *SAS Viya Formats and Informats: Reference*
NENGO Informat
Reads Japanese date values in the form eyymmd.

Category: Date and Time

Syntax
NENGOw:

Syntax Description
w
specifies the width of the input field.

Default 10
Range 7–32

Details
The general form of a Japanese date is eyymmd:

- e is the first letter of the name of the imperial era (Meiji, Taisho, Showa, or Heisei).
- 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 examples use different input values.

```sas
input nengo_date nengo8.;
put nengo_date date9.;
data _null_
  input nengo_date nengo8.;
  put nengo_date date9.;
  put nengo_date=;
datalines;
h11108
h.11108
11/10/08
```
See Also

Formats:

- “NENGO Format” on page 102

**NLDATE Informat**

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

**Category:** Date and Time  
**Alias:** NLDATEW

**Syntax**

NLDATE.w.

**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>h11108</td>
<td>08OCT1999</td>
</tr>
<tr>
<td>h.11108</td>
<td>08OCT1999</td>
</tr>
<tr>
<td>11/10/08</td>
<td>08OCT1999</td>
</tr>
</tbody>
</table>
See Also

Format:

- “NLDATE Format” on page 106

NLDATEW 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.

Category: Date and Time

Alignment: Left

Syntax

NLDATEWw.

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

**Category:** Date and Time  
**Alias:** NLDATMW, NLDATMAP

**Syntax**

\[ \text{NLDATM} w. \]

**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></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,nldatm200.);</td>
<td>19778</td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></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>

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>y=input('24.Feb03:12:39:43', nldatm.);</td>
<td>1361709583</td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>
### 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.

- **Category:** Date and Time
- **Alignment:** Left

### Syntax

\[ \text{NLDATMAP}_w. \]

### Syntax Description

- \( w \) specifies the width of the output field. If necessary, SAS abbreviates the datetime value to fit the format width.

  - **Default:** 32
  - **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>
| options locale=German_Germany;  
y=input('24.Februar 2003 12.39 Uhr', nldatm.);  
put y=; | 1330171200 |

---

See Also

Format:
- “NLDATM Format” on page 127
Statements | Results
---|---
options locale=Spanish_Mexico;
data;
  dy='24/02/2003 12:39:43 PM';
y=input(dy,nldatmap200.);
  put y=;
run;
1708864783

**NLDATMW 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.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

\[ \text{NLDATMW}_w \]

**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.

Statements | Results
---|---
options locale=English_UnitedStates;
data;
  dy='Mon, Feb 24, 2014 12:39:43 PM';
y=input(dy,nldatmw200.);
  put y=;
run;
1708864783

options locale=German_Germany;
  dy='Mo, 24. Feb 2014 12.39 Uhr';
y=input(dy,nldatmw16.);
  put y=;
run;
1708864783
**NLMNIAED Informat**

Reads the monetary format of the international expression for the United Arab Emirates.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNIAED}w.d \]

**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:** 3
- **Range:** 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmniaed32.2);
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:**

- “NLMNLAED Informat” on page 492
NLMNIAUD Informat

Reads the monetary format of the international expression for Australia.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNIAUD}_{w.d}

\textit{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 \(d\) 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)',nlmiaud32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

\begin{tabular}{|c|c|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=;} & -12345.67 \\
\texttt{put y=;} & -12345.67 \\
\hline
\end{tabular}

See Also

Informat:

- “NLMNLAUD Informat” on page 493
NLMNIBGN Informat

Reads the monetary format of the international expression for Bulgaria.

**Category:** Numeric
**Alignment:** Left

---

**Syntax**

\texttt{NLMNIBGNw.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=put('(-1234.56789)',nlmnibgn32.2);
y=put('(-1234.56789)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>--------------</td>
</tr>
<tr>
<td>put \texttt{x=};</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put \texttt{y=};</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

**See Also**

- “NLMNLBGN Informat” on page 494
NLMNIBRL Informat
Reads the monetary format of the international expression for Brazil.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\texttt{NLMNIBRL\_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 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 \texttt{x=};</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put \texttt{y=};</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

**See Also**

**Informat:**

- “NLMNLBRL Informat” on page 495
**NLMNICAD Informat**

Reads the monetary format of the international expression for Canada.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

**Syntax**

\[ \text{NLMNICAD}w.d \]

**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`.

\[ \begin{align*}
  x &= \text{input}('($12,345.67)\', \text{nlmnicad32.2}); \\
  y &= \text{input}('($12,345.67)\', \text{dollar32.2});
\end{align*} \]

<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:**

- “NLMNICAD Format” on page 156
NLMNICHF Informat

Reads the monetary format of the international expression for Liechtenstein and Switzerland.

**Category:** 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.

```plaintext
x=input'($12,345.67)',nlmnichf32.2);
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:**

- “NLMNICHF Format” on page 157
NLMNICNY Informat

Reads the monetary format of the international expression for China.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

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.

\(x=\)input\(\cdot\)'($\text{\$12,345.67}\',\text{\$12,345.67};n\text{\$12,345.67})\);
\(y=\)input\(\cdot\)'($\text{\$12,345.67}\',\text{\$12,345.67};\text{\$12,345.67})\);

<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:**

- “NLMNICNY Format” on page 158
NLMNICZK Informat
Reads the monetary format of the international expression for the Czech Republic.

Category: Numeric
Alignment: Left

Syntax
NLMNICZKw.d

Syntax Description

\textbf{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{d} specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0–28</td>
</tr>
</tbody>
</table>

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=input('($12,345.67)',nlmniczk32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----+</td>
<td></td>
</tr>
<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:}

- “NLMNLCZK Informat” on page 499
**NLMNIDKK Informat**

Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNIDKK} \, w \cdot d \]

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 912
- **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)',nlmndkk32.2);
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:**

- “NLMNIDKK Format” on page 160
**NLMNIEEK Informat**

Reads the monetary format of the international expression for Estonia.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNIEEK} w.d \]

### 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)',nlmniek32.2);
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:**
- “NLMNLEEK Informat” on page 501
NLMNIEGP Informat

Reads the monetary format of the international expression for Egypt.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[
\text{NLMNIEGP} \ w \ . \ d
\]

### 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:** 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:**
- “NLMNLEGP Informat” on page 502
NLMNIEUR 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.

Category: 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.

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:

- “NLMNIEUR Format” on page 163
NLMNIGBP Informat

Reads the monetary format of the international expression for the United Kingdom.

Category: Numeric
Alignment: Left

Syntax

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 LOCALE= system option is set to English_UnitedStates.

\texttt{x= input' \$(12,345.67)' , nlmnigbp32.2);}
\texttt{y= input' \$(12,345.67)'dollar32.2);}

<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

Format:

- “NLMNIGBP Format” on page 164
NLMNIHKD Informat

Reads the monetary format of the international expression for Hong Kong.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNIHKD<sup>_w.d_</sup>

**Syntax Description**

<sup>_w_</sup>

specifies the width of the output field.

- **Default:** 12  
- **Range:** 8–32

<sup>_d_</sup>

specifies to divide the number by 10<sup>_d_</sup>. 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)',nlmnihkd32.2);
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:**

- “NLMNIHKD Format” on page 165
NLMNIHRK Informat

Reads the monetary format of the international expression for Croatia.

**Category:** Numeric

**Alignment:** Left

---

**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_.

```plaintext
x=input('($12,345.67)', nlmnihrk32.2);
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:**

- “NLMNLHRK Informat” on page 506
NLMNIHUF Informat

Reads the monetary format of the international expression for Hungary.

**Category:** 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.

```plaintext
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:**

- “NLMNLHUF Informat” on page 507
**NLMNIIDR Informat**

Reads the monetary format of the international expression for Indonesia.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[ \text{NLMNIIDR} \, \text{w.d} \]

### Syntax Description

\( \text{w} \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( \text{d} \)

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:**

- “NLMNLIDR Informat” on page 508
NLMNIILS Informat

Reads the monetary format of the international expression for Israel.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[
NLMNIILS w.d
\]

**Syntax Description**

\[w\]

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>8–32</td>
</tr>
</tbody>
</table>

\[d\]

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

<table>
<thead>
<tr>
<th>Default</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>0–28</td>
</tr>
</tbody>
</table>

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{input}'($12,345.67)',\text{nlnmniils32.2);} \\
y &= \text{input}'($12,345.67)'dollar32.2); \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+----1----+</td>
<td></td>
</tr>
<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:**

- “NLMNIILS Format” on page 169
NLMNIINR Informat

Reads the monetary format of the international expression for India.

Category: Numeric
Alignment: Left

Syntax

NLMNIINR 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('($12,345.67)',nlmniinr32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=</td>
<td></td>
</tr>
<tr>
<td>y=</td>
<td></td>
</tr>
<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:
  • “NLMNLINR Informat” on page 510
**NLMNIJPY Informat**

Reads 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 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)',nlmnijpy32.2);
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

- “NLMNIJPY Format” on page 171
**NLMNIKRW Informat**

Reads the monetary format of the international expression for South Korea.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNIKRW}w.d \]

### 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 LOCATE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)','nlmnikrw32.2);
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:**
  - “NLMNLKRW Informat” on page 512
NLMNILTL Informat

Reads the monetary format of the international expression for Lithuania.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNILTL} w.d \]

**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 \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
x=input('($12,345.67)',nlmniltl32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
</tr>
<tr>
<td>put y=;</td>
</tr>
</tbody>
</table>

**Results**

\begin{verbatim}
-12345.67
-12345.67
\end{verbatim}

**See Also**

**Informat:**

- “NLMNLLTL Informat” on page 513
**NLMNILVL Informat**

Reads the monetary format of the international expression for Latvia.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\[ \text{NLMNILVL}w.d \]

**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.

\[
\begin{align*}
x &= \text{input}('($12,345.67)' , \text{nlmnilv132.2}); \\
y &= \text{input}('($12,345.67)' , \text{dollar32.2}); \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-12345.67</td>
</tr>
<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:**
- “NLMNLLVL Informat” on page 514
**NLMNIMOP Informat**

Reads the monetary format of the international expression for Macau.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\[ \text{NLMNIMOP}\, w.d \]

**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.

\[
\begin{align*}
\text{x}=\text{input(’($12,345.67)$’,nlmnimop32.2)}; \\
\text{y}=\text{input(’($12,345.67)$’,dollar32.2)};
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \text{x} =;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put \text{y} =;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

**See Also**

**Informat:**

- “NLMNLMOP Informat” on page 515
**NLNMIMXN Informat**

Reads the monetary format of the international expression for Mexico.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLNMIMXN} w.d \]

**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)',nlnmimxn32.2);
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**

- “NLNMNLXN Informat” on page 516
**NLMNIMYR Informat**

Reads the monetary format of the international expression for Malaysia.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

NLMNIMYRw.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)',nlnmiry32.2);
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:**

- “NLMNIMYR Format” on page 177
NLMNINOK Informat

Reads the monetary format of the international expression for Norway.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNINOK} \_w.d \]

**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)','nlmminok32.2');
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:**

- “NLMNINOK Format” on page 178
**NLMNINZD Informat**

Reads the monetary format of the international expression for New Zealand.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

$\text{NLMNINZD}w.d$

**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.

```plaintext
x=input('($12,345.67)',nlnminzd32.2);
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:**

- “NLMNINZD Format” on page 179
NLMNIPLN Informat

Reads the monetary format of the international expression for Poland.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNIPLNW.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)',nlmnipln32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----+----------</td>
<td></td>
</tr>
<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:**

- “NLMNIPLN Format” on page 180
**NLMNIRUB Informat**

Reads the monetary format of the international expression for Russia.

- **Category:** Numeric
- **Alignment:** Left

## Syntax

\[ \text{NLMNIRUB} \ w.d \]

### 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)',nlmnirub32.2);
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

- [“NLMNIRUB Format” on page 181](#)
**NLMNISEK Informat**

Reads the monetary format of the international expression for Sweden.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[ \text{NLMNISEK} \_w.d \]

### 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)',nlnmisek32.2);
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:**

- [“NLMNISEK Format” on page 182](#)
NLMNISGD Informat

Reads the monetary format of the international expression for Singapore.

Category: Numeric
Alignment: Left

Syntax

NLMNISGDw.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)',nlmnisgd32.2);
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:
- “NLMNISGD Format” on page 183
NLNMITHB Informat

Reads the monetary format of the international expression for Thailand.

Category: Numeric
Alignment: Left

Syntax

NLNMITHBw.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)',nlmithb32.2);
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:

• “NLMLTHB Informat” on page 524
**NLMNITRY Informat**

Reads the monetary format of the international expression for Turkey.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

```
NLMNITRY 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=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><code>put x=</code>;</td>
<td>-12345.67</td>
</tr>
<tr>
<td><code>put y=</code>;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

#### See Also

- **‘NLMNLTRY Informat” on page 525**
NLMNITWD Informat

Reads the monetary format of the international expression for Taiwan.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNITWD\[w,d\]

**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)',nlmnitwd32.2);
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:**
  - “NLMNITWD Format” on page 186
**NLMNIUSD Informat**

Reads the monetary format of the international expression for Puerto Rico and the United States.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNIUSD} w.d \]

**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)',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:**

- “NLMNIUSD Format” on page 187
NLMNIZAR Informat

Reads the monetary format of the international expression for South Africa.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNIZAR$w.d$

**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'($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:**

- “NLMNIZAR Format” on page 188
**NLMNLAED Informat**
Reads the monetary format of the local expression for the United Arab Emirates.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLAED} w.d \]

**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:** 3
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{input}('($12,345.67')',\text{nlnlaxed32.2}); \\
y &= \text{input}('($12,345.67')',\text{dollar32.2});
\end{align*}
\]

<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:**

- “NLMNIAED Informat” on page 455
NLMNLAUD Informat

Reads the monetary format of the local expression for Australia.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLAUD}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 \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
x=input('$12,345.67',nlmnlaud32.2);
y=input('$12,345.67',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| \begin{verbatim}
put x=;
\end{verbatim} | -12345.67 |
| \begin{verbatim}
put y=;
\end{verbatim} | -12345.67 |

See Also

Format:

- “NLMNLAUD Format” on page 190
NLMNLBGN Informat
Reads the monetary format of the local expression for Bulgaria.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\texttt{NLMNLBGN}w.d

**Syntax Description**

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

\textit{Default} 12  
\textit{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.

\textit{Default} 2  
\textit{Range} 0–28

---

**Example**

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English_UnitedStates}.

\begin{verbatim}
x=input(-12345.67,nlmnlbgn32.2);
y=input(-12345.67,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
<td>-12345.67</td>
</tr>
<tr>
<td>( )</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

**See Also**

**Informat:**

- “NLMNIBGN Informat” on page 457
NLMNLBRL Informat
Reads the monetary format of the local expression for Brazil.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLBRL} 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 \texttt{LOCALE} system option is set to English_UnitedStates.

\begin{verbatim}
x=input('($12,345.67)',nlmnlbrl32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Statements & Results \\
\hline
\texttt{put x=} & -12345.67 \\
\texttt{put y=} & -12345.67 \\
\hline
\end{tabular}
\end{table}

See Also

Informat:
- “NLMNIBRL Informat” on page 458
NLMNLCAD Informat

Reads the monetary format of the local expression for Canada.

Category: 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.

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:

- “NLMNLCAD Format” on page 193
NLMNLCHF Informat
Reads the monetary format of the local expression for Liechtenstein and Switzerland.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLCHFw.d}

\textbf{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

\textbf{Example}

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
x=input('($12,345.67)',nlmnlchf32.2);
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}

\textbf{See Also}

Format:
- “NLMNLCHF Format” on page 194
**NLMNL\text{CNY} Informat**

Reads the monetary format of the local expression for China.

- **Category:** Numeric
- **Alignment:** Left

---

**Syntax**

\texttt{NLMNL\text{CNY}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 English_UnitedStates.

\begin{verbatim}
x=input('(''(\$12,345.67)'',nlmlncny32.2);
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:**

- “NLMNL\text{CNY} Format” on page 195
NLMNLCZK Informat
Reads the monetary format of the local expression for the Czech Republic.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLCZK} \ w.d \]

**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 \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
x=input('($12,345.67)',nlmnlczk32.2);
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**

**Informat:**

- “NLMNICZK Informat” on page 462
**NLMNLDKK Informat**
Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNLDKKw.d} \]

**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)',nlmnldkk32.2);
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**

- “NLMNLDKK Format” on page 197
NLMNLEEK Informat
Reads the monetary format of the local expression for Estonia.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLEEK\textit{w,d}}

\textit{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 \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
x=input('($12,345.67)',nlmnleek32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-12345.67</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNIEEK Informat” on page 464
NLMNLEGP Informat

Reads the monetary format of the local expression for Egypt.

Category: Numeric
Alignment: Left

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.

``` Sas
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:
- “NLMNIEGP Informat” on page 465
NLMNLEUR Informat

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.

Category: Numeric
Alignment: Left

Syntax

NLMNLEUR 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('($12,345.67)',nlmleur32.2);
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:

- “NLMNLEUR Format” on page 200
NLMNLGBP Informat
Reads the monetary format of the local expression for the United Kingdom.

Category: Numeric
Alignment: Left

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.

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>x=</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:
• "NLMNLGBP Format" on page 201
NLMNLHKD Informat

Reads the monetary format of the local expression for Hong Kong.

- **Category:** Numeric
- **Alignment:** Left

### 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.

```
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

- “NLMNLHKD Format” on page 202
NLMNLHRK Informat
Reads the monetary format of the local expression for Croatia.

Category: Numeric
Alignment: Left

Syntax

\texttt{\textsc{NLMNLHRK}w,d}

\textbf{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)',nlmnlhrk32.2);
y=input(' ($12,345.67)',dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \texttt{x=};</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put \texttt{y=};</td>
<td>-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Informat:
- “NLMNIHRK Informat” on page 469
NLMNLHUF Informat
Reads the monetary format of the local expression for Hungary.

Category: Numeric
Alignment: Left

Syntax

NLMNLHUFw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 12
Range \( 8 \text{–} 32 \)

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

Default 2
Range \( 0 \text{–} 28 \)

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x & = \text{input}('($12,345.67)' \text{, nlmnlhuf32.2}); \\
y & = \text{input}('($12,345.67)' \text{, dollar32.2});
\end{align*}
\]

<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:

- “NLMNIHUF Informat” on page 470
NLMNLIDR Informat
Reads the monetary format of the local expression for Indonesia.

Category: Numeric
Alignment: Left

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.

\[ x = \text{input}('($12,345.67)', nlmnlidr32.2); \]
\[ y = \text{input}('($12,345.67)', dollar32.2); \]

<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

Informat:
- “NLMNIIDR Informat” on page 471
NLMNLILS Informat

Reads the monetary format of the local expression for Israel.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLILS}w.d

\textit{Syntax Description}

\textit{w}

specifies the width of the output field.

Default \hfill 12

Range \hfill 8–32

\textit{d}

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default \hfill 4

Range \hfill 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.
\begin{verbatim}
x=input('($12,345.67)',nlmnlils32.2);
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:

\begin{itemize}
\item \texttt{“NLMNLILS Format” on page 206}
\end{itemize}
NLMNLINR Informat

Reads the monetary format of the local expression for India.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

NLMNLINRw.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)',nlmlinr32.2);
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**

- “NLMNIINR Informat” on page 473
NLMNLJPY Informat

Reads the monetary format of the local expression for Japan.

Category:  Numeric
Alignment:  Left

Syntax

NLMNLJPYw.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)',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 Format” on page 208
**NLMNLKRW Informat**

Reads the monetary format of the local expression for South Korea.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNLKRW}w.d \]

**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)',nlmnlkrw32.2);
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**

- “NLMNIKRW Informat” on page 475
NLMNLLTL Informat

Reads the monetary format of the local expression for Lithuania.

Category: Numeric
Alignment: Left

Syntax

NLMNLLTLw.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=input('($12,345.67)',nlnlltl32.2);
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:

- “NLMNILTL Informat” on page 476
NLMNLLVL Informat
Reads the monetary format of the local expression for Latvia.

**Category:** Numeric  
**Alignment:** Left

**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 Informat” on page 477
NLMNLMOP Informat

Reads the monetary format of the local expression for Macau.

Category: Numeric
Alignment: Left

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:
- “NLMNIMOP Informat” on page 478
NLMNLMXN Informat

Reads the monetary format of the local expression for Mexico.

Category: Numeric
Alignment: Left

Syntax

NLMNLMXN\(w.d\)

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.

\[x = \text{input}('(\$12,345.67)', \text{nlnlmxn32.2});\]
\[y = \text{input}('(\$12,345.67)', \text{dollar32.2});\]

<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:

- “NLMNLMXN Informat” on page 479
**NLMNLMYR Informat**

Reads the monetary format of the local expression for Malaysia.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLMYR} \ w.d \]

**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)',nlmnlmyr32.2);
y=input('($12,345.67)',dollar32.2);

Statements | Results
---|---
| | ----+----1----+
| put x=; | -12345.67
| put y=; | -12345.67
```

---

**See Also**

**Format:**

- “NLMNLMYR Format” on page 214
NLMLNOK Informat
Reads the monetary format of the local expression for Norway.

**Category:** Numeric  
**Alignment:** Left

---

### Syntax

\[ \text{NLMLNOK} w.d \]

### 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 \texttt{LOCALE=} system option is set to English UnitedStates.

\[
\text{x=input('($12,345.67)',nlmnlno32.2);} \\
\text{y=input('($12,345.67)',dollar32.2);} \\
\]

<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:**
- “NLMLNOK Format” on page 215
NLMNLNZD Informat
Reads the monetary format of the local expression for New Zealand.

Category: Numeric
Alignment: Left

Syntax

\[ \text{NLMNLNZD} \, w.d \]

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`.

\[ \text{x=input}'(\$(12,345.67)',\text{nlnlnzd32.2}); \]
\[ \text{y=input}'(\$(12,345.67)',\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put x=;}</td>
<td>-12345.67</td>
</tr>
<tr>
<td>\text{put y=;}</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLNZD Format” on page 216
NLMNLPLN Informat

Reads the monetary format of the local expression for Poland.

Category: Numeric
Alignment: Left

Syntax

NLMNLPLNW.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)',nlmnlpln32.2);
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:

• “NLMNLPLN Format” on page 217
NLMNLRUB Informat

Reads the monetary format of the local expression for Russia.

Category: Numeric
Alignment: Left

Syntax

NLMNLRUBw.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)',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:
- “NLMNLRUB Format” on page 218
NLMNLSEK Informat

Reads the monetary format of the local expression for Sweden.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNLSEK}w.d \]

### 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)',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

- [“NLMNLSEK Format” on page 219](#)
NLMNLSGD Informat

Reads the monetary format of the local expression for Singapore.

Category: Numeric
Alignment: Left

Syntax

\text{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)’,nlmlsgd32.2);
y=input(’($12,345.67)’,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLSGD Format” on page 220
NLMNLTHB Informat

Reads the monetary format of the local expression for Thailand.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLTHB\(w,d\)

**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)',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:**

- “NLMNITHB Informat” on page 487
NLMNLTRY Informat

Reads the monetary format of the local expression for Turkey.

Category: Numeric
Alignment: Left

Syntax

NLMNLTRY+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)',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:

- “NLMNITRY Informat” on page 488
**NLMNL TWD Informat**

Reads the monetary format of the local expression for Taiwan.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\[ \texttt{NLMNL TWD}w.d \]

**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)',nlmnltdw32.2);
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:**

- “NLMNL TWD Format” on page 223
NLMNLUSD Informat

Reads the monetary format of the local expression for Puerto Rico, and the United States.

**Category:** Numeric  
**Alignment:** Left

## Syntax

\[ \text{NLMNLUSD}_{w,d} \]

### 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)',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:**

- “NLMNLUSD Format” on page 224
NLMNLZAR Informat

Reads the monetary format of the local expression for South Africa.

Category: Numeric
Alignment: Left

Syntax

NLMNLZAR w.d

Syntax Description

w

specifies the width of the output field.

Default: 12
Range: 8–32

d

specifies to divide the number by 10d. 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)',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:
- “NLMNLZAR Format” on page 225
NLMNY Informat

Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value.

Category: 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.

data;
x=input('1', NLMNY32.);
put x=;
run;

Example

The following examples use the input value of $12,345.67.
### Statements

<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:**
- “NLMNY Format” on page 226
- “NLMNYI Format” on page 227

**Informat:**
- “NLMNYI Informat” on page 530

---

### NLMNYI Informat

Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

\[ \text{NLMNYI} \_w.d \]

**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 NLMNYIw.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 NLMNYIw.d informat performs processing that is the opposite of the NLMNYw.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:
- “NLMNY Format” on page 226
- “NLMNYI Format” on page 227

Informat:
- “NLMNY Informat” on page 529

NLNUM Informat

Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.

Category: Numeric

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 \(10^d\). If the data contains decimal separators, the \( d \) value is ignored.

Default: 0

Range: 0–31

Details

The NLNUM\(w,d\) 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:

```plaintext
data;
x=input('1',NLNUM32.);
put x=;
run;
```

Comparisons

The NLNUM\(w,d\) informat performs processing that is opposite to the NLNUMI\(w,d\) 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></td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>(-1234356.78)</td>
</tr>
</tbody>
</table>

```plaintext
data;
options locale=English_UnitedStates;
x=input(-1,234,356.78',nlnum32.2);
put x=;
run;
```

See Also

Formats:

- “NLNUM Format” on page 228
NLNUMI Informat

Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

```plaintext
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 NLNUMI $w.d$ 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 NLNUMI $w.d$ informat performs processing that is opposite of the NLNUM $w.d$ 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></td>
<td>$-1.23435678$</td>
</tr>
</tbody>
</table>
Statements | Results
---|---
```python
options locale=English_UnitedStates;
x=input('-1,234,356.78', nlnumi32.2);
put x=;
```

-1234356.78

**See Also**

Formats:
- “NLNUM Format” on page 228
- “NLNUMI Format” on page 230
- “NLNUM Informat” on page 531

**NLPCT Informat**

Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

```
NLPCT w.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\textit{w.d} informat performs processing that is opposite of the NLPCTI\textit{w.d} informat. The NLPCT\textit{w.d} informat is similar to the PERCENT\textit{w.d} informat except that the NLPCT\textit{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.

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;</td>
<td></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>(-123.4567)</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- “NLPCT Format” on page 231
- “NLPCTI Format” on page 232
- “NLPCTI Informat” on page 535

NLPCTI Informat

Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.

Category: Numeric

Syntax

\texttt{NLPCTI\textit{w.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 NLPCTI\(w,d\) 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 NLPCTI\(w,d\) informat performs processing that is opposite of the NLPCT\(w,d\) 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>-123.4567</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- “NLPCT Format” on page 231
- “NLPCTI Format” on page 232

Informat:

- “NLPCT Informat” on page 534
NLTimap 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.

**Category:** Date and Time

**Syntax**

```
NLTIMAP_w.
```

**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>---+----1----+</td>
<td></td>
</tr>
<tr>
<td><code>options locale=English_UnitedStates;</code></td>
<td>16:24:43</td>
</tr>
<tr>
<td><code>y=input('04:24:43 PM',nltimap11.);</code></td>
<td></td>
</tr>
<tr>
<td><code>put y time.;</code></td>
<td></td>
</tr>
<tr>
<td><code>options locale=German_Germany;</code></td>
<td>16:24:00</td>
</tr>
<tr>
<td><code>y=input('16.24 Uhr',nltimap11.);</code></td>
<td></td>
</tr>
<tr>
<td><code>put y time.;</code></td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “NLTIMAP Format” on page 241

NLTime Informat

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

**Category:** Date and Time
Alias: NLTIMAP

Syntax

NLTIME\(w\).

Syntax Description

\(w\) specifies the width of the input field.

Default 20

Range 10–200

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;</td>
<td></td>
</tr>
<tr>
<td>y=input('16:24:43',nltime.);</td>
<td>16:24:43</td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>16:24:00</td>
</tr>
<tr>
<td>y=input('16.24 Uhr',nltime.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLTIME Format” on page 242

$UCS2B 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

Syntax

$UCS2B\(w\):
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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('5927'x,$ucs2b.);</td>
<td>x=E5A4A7</td>
</tr>
<tr>
<td>put x=$hex6.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “\$UCS2B Format” on page 243
- “\$UCS2L Format” on page 245
- “\$UCS2X Format” on page 248
- “\$UTF8X Format” on page 264

Informats:
- “\$UCS2L Informat” on page 540
- “\$UCS2X Informat” on page 543
- “\$UTF8X Informat” on page 557

\$UCS2BE 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

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 2–32767

Comparisons

The \$UCS2BEw. informat performs processing that is opposite of the \$UCS2Bw. informat.

Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
</table>
| ucs2str=input ('?

$ucs2be4.');
| ucs2str=00205927 |
| put ucs2str=$hex8.; |

See Also

Formats:

• “\$UCS2B Format” on page 243
• “\$UCS2BE Format” on page 244

Informat:

• “\$UCS2B Informat” on page 538

\$UCS2L 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('2759'x,$ucs2l.); x=$hex6.;</td>
<td>E5A4A7</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2B Format” on page 243
- “$UCS2L Format” on page 245
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 264

Informats:
- “$UCS2B Informat” on page 538
- “$UCS2X Informat” on page 543
- “$UTF8X Informat” on page 557
$UCS2LE 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input ('ä', $ucs2le4.);</td>
<td>ucs2str=2759</td>
</tr>
<tr>
<td>put ucs2str=$hex4;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2L Format” on page 245
- “$UCS2LE Format” on page 247

Informat:
- “$UCS2L Informat” on page 540
$UCS2X 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:** UTF-8 is the only SAS session encoding supported by SAS Viya.

**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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('5927'x,$ucs2x.);</td>
<td>x=E29D99</td>
</tr>
<tr>
<td>put x=$hex6.;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Formats:

- “$UCS2B Format” on page 243
- “$UCS2L Format” on page 245
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 264
$UCS2XE 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:** UTF-8 is the only SAS session encoding supported by SAS Viya.

**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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>------------------</td>
</tr>
<tr>
<td>ucs2str=input (’э’, $ucs2xe4.);</td>
<td>ucs2str=20002759</td>
</tr>
<tr>
<td>put ucs2str=$hex8;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

• “$UCS2X Format” on page 248
• “$UCS2XE Format” on page 249

Informat:
• “$UCS2X Informat” on page 543

$UCS4B 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

Syntax
$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 $UCS4Xw. informat. If you are processing data from different operating environments, then use the $UCS4Bw. and $UCS4Lw. informats.

Example
These examples use UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs4str=input('á', $ucs4be8.);</td>
<td>ucs4str=0000002000005927</td>
</tr>
<tr>
<td>put ucs4str= $hex16.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also
Format:
$UCS4L 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:** UTF-8 is the only SAS session encoding supported by SAS Viya.

**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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>z=put('.com',$UCS4L16.);</td>
<td>2E0000006300000006F0000006D000000</td>
</tr>
<tr>
<td>put z $hex32.;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Format:**
$UCS4X 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:** UTF-8 is the only SAS session encoding supported by SAS Viya.

**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 UTF-8 encoding and little-endian formatting.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>uc4s=put('91e5'x,$ucs4x.);</td>
<td>uc4s=FDF00000</td>
</tr>
<tr>
<td>utf8=input(uc4s,$ucs4x.);</td>
<td>utf8=EFBFBD20</td>
</tr>
<tr>
<td>put uc4s=$hex8. utf8=$hex8.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:
- “$UCS2X Format” on page 248
- “$UCS2B Format” on page 243
- “$UCS2L Format” on page 245
- “$UCS4X Format” on page 255
- “$UTF8X Format” on page 264

Informats:
- “$UCS2B Informat” on page 538
- “$UCS2L Informat” on page 540
- “$UTF8X Informat” on page 557

$UCS4XE 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

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 UTF-8 encoding.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ucs4xe4('%', $ucs4xe8.); put ucs4str=$hex16;</td>
<td>ucs4str=2000000027590000</td>
</tr>
</tbody>
</table>

See Also

Formats:
- "$UCS4X Format" on page 255
- "$UCS4XE Format" on page 256

Informat:
- "$UCS4X Informat" on page 547

$UESC 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

Syntax

$UESCw.

Syntax Description

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 $UESCw. informat can be nested.

Comparisons

The $UESCw. informat performs processing that is the opposite of the $UESCEw. informat.
Example

These examples use the UTF-8 encoding, which is supported under the Linux operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('{/u5927}', $uesc10.);</td>
<td>\u5927</td>
</tr>
<tr>
<td>y=input('{\u5927}', $uesc10.);</td>
<td>\u5927</td>
</tr>
<tr>
<td>z=input('{\uu5927}', $uesc10.);</td>
<td>\uu5927</td>
</tr>
<tr>
<td>put x; put y; put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UESC Format” on page 257
- “$UESCE Format” on page 259

Informat:
- “$UESCE Informat” on page 550

$UESCE 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: UTF-8 is the only SAS session encoding supported by SAS Viya.

Syntax

$UESCE\textsubscript{w}.

Syntax Description

\textit{w}

specifies the width of the input field.

Default \begin{align*} 8 \end{align*}

Range \begin{align*} 1–32767 \end{align*}

Details

The $UESCE\textsubscript{w}$ 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 Linux operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('¥', $uescel0.);</td>
<td>\u5927</td>
</tr>
<tr>
<td>y=input('¥u5927', $uescel0.);</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- “$UESC Format” on page 257
- “$UESCE Format” on page 259

Informat:

- “$UESC Informat” on page 549

$UNCR Informat

Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.

Category: Character

Restriction: UTF-8 is the only SAS session encoding supported by SAS Viya.

Syntax

$UNCRw:

**Syntax Description**

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 UTF-8 encoding.

<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:
- “$UNCR Format” on page 260
- “$UNCRE Format” on page 261

Informat:
- “$UNCRE Informat” on page 552

$UNCRE Informat
Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.

Category: Character
Restriction: UTF-8 is the only SAS session encoding supported by SAS Viya.

Syntax
$UNCReW.
Syntax Description

\( w \)

specifies the width of the input field.

Default 8

Range 1–32767

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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---+--------+1++---+</td>
</tr>
<tr>
<td>x=input ('\x abc', $uncrel2.);</td>
<td>&amp;#22823;abc</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UNCR Format” on page 260
- “$UNCRE Format” on page 261

Informat:
- “$UNCR Informat” on page 551

$UPAREN 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: UTF-8 is the only SAS session encoding supported by SAS Viya.
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 UTF-8 encoding.

<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>五</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:

- “$UPAREN Format” on page 262
- “$UPARENE Format” on page 263

Informats:

- “$UPARENE Informat” on page 555
- “$UPARENP Informat” on page 556
$UPAREN$ 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:** UTF-8 is the only SAS session encoding supported by SAS Viya.

### Syntax

$SUPAREN_w$.

### Syntax Description

`w` specifies the width of the input field.

**Default:** 8

**Range:** 1–32767

### Comparisons

The $SUPAREN_w$. informat performs processing that is opposite of the $SUPAREN_w$. informat.

### Example

These examples use UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>v=input('a',$uparen10.);</code></td>
<td>&lt;u0061&gt;</td>
</tr>
<tr>
<td><code>w=input('b',$uparen10.);</code></td>
<td>&lt;u0062&gt;</td>
</tr>
<tr>
<td><code>x=input('c',$uparen10.);</code></td>
<td>&lt;u0063&gt;</td>
</tr>
<tr>
<td><code>y=input('3',$uparen10.);</code></td>
<td>&lt;u0033&gt;</td>
</tr>
<tr>
<td><code>z=input('Actualizar',$uparen10.);</code></td>
<td>&lt;u5927&gt;</td>
</tr>
<tr>
<td><code>put v;</code></td>
<td></td>
</tr>
<tr>
<td><code>put w;</code></td>
<td></td>
</tr>
<tr>
<td><code>put x;</code></td>
<td></td>
</tr>
<tr>
<td><code>put y;</code></td>
<td></td>
</tr>
<tr>
<td><code>put z;</code></td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Formats:**

- “$SUPAREN$ Format” on page 262
$UPAREN 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:** UTF-8 is the only SAS session encoding supported by SAS Viya.

**Syntax**

$UPAREN P

**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 UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('&lt;u0061&gt;',$uparenp10.);</td>
<td>a</td>
</tr>
<tr>
<td>w=input('&lt;u0062&gt;',$uparenp10.);</td>
<td>b</td>
</tr>
<tr>
<td>x=input('&lt;u0063&gt;',$uparenp10.);</td>
<td>c</td>
</tr>
<tr>
<td>y=input('&lt;u0033&gt;',$uparenp10.);</td>
<td>3</td>
</tr>
<tr>
<td>z=input('&lt;u5927&gt;',$uparenpnp10.);</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>
$UTF8X 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

Syntax

$UTF8Xw.

Syntax Description

w
specifies the width of the input field.

Default 8
Range 1–32767

Comparisons

This example uses the Japanese UTF-8 encoding, which is supported under the Linux operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('e5a4a7'x, $utf8x3.); put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

• “$UCS2B Format” on page 243
Informats:

- “$UCS2B Informat” on page 538
- “$UCS2L Informat” on page 540
- “$UCS2X Informat” on page 543
Part 7

Macro Functions for NLS

Chapter 14

Dictionary of Macro Functions for NLS   561
Chapter 14
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 and %QKCMPRES Macro Function (p. 562)</td>
<td>Compresses multiple blanks and removes leading and trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%KINDEX Macro Function (p. 562)</td>
<td>Returns the position of the first character of a string.</td>
</tr>
<tr>
<td></td>
<td>%KLEFT and %QKLEFT Macro Functions (p. 563)</td>
<td>Left-aligns an argument by removing leading blanks.</td>
</tr>
<tr>
<td></td>
<td>%KLENGTH Macro Function (p. 563)</td>
<td>Returns the length of a string.</td>
</tr>
<tr>
<td></td>
<td>%KSCAN and %QKSCAN Functions (p. 564)</td>
<td>Search for a word that is specified by its position in a string.</td>
</tr>
<tr>
<td></td>
<td>%KSUBSTR and %QKSUBSTR Macro Functions (p. 566)</td>
<td>Produce a substring of a character string.</td>
</tr>
</tbody>
</table>
%KUPCASE and %QKUPCASE Macro Functions (p. 568)

Convert values to uppercase.

---

**Dictionary**

**%KCMPRES and %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 and %QKLEFT Macro Functions**

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:

```sas
%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 and \%QKSCAN Functions

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 *n*th 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:

```plaintext
& % ' " ( ) + - * / < > = ¬ ^ ~ ; , # blank  
AND OR NOT EQ 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:
  ```plaintext
  blank ! $ % & ( ) * + - . / ; < ^ | ¦
  ```
  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:
  ```plaintext
  blank ! $ % & ( ) * + - . / ; < ¬ | ¦
  ```

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:

```plaintext
$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 and %QKSUBSTR Macro Functions

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.

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.
%QKSUBSTR 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:
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: mestic operand is required. A character
Message 4 is: operand was found in the %EVAL function
Message 5 is: or %IF condition where a numeric operan
Message 6 is: d 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:

```sas
%let a=one;
%let b=two;
%let c=%nrstr(&a &b);
%put C: &c;
%put With KSUBSTR: %ksubstr(&c,1,2);
%put With QSUBSTR: %qKsubstr(&c,1,2);
```

When these statements execute, these lines are written to the SAS log:

```
C: &a &b
With KSUBSTR: one
With QSUBSTR: &a
```

%KUPCASE and %QKUPCASE Macro Functions

Convert values to uppercase.

**Category:** DBCS  
**Type:** NLS macro function

**Syntax**

```sas
%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 FSVIEW 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
Part 8

System Options for NLS

Chapter 15

*Dictionary of System Options for NLS* ............ 573
The language control category of SAS system options is 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: Language Control</td>
<td>DFLANG= System Option (p. 575)</td>
<td>Specifies the language for international date informats and formats.</td>
</tr>
<tr>
<td>ENC encoding</td>
<td>ENC encoding</td>
<td>Specifies the default character-set encoding for the SAS session.</td>
</tr>
<tr>
<td>LOCALE System Option (p. 578)</td>
<td>LOCALE System Option (p. 578)</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>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>LOCALEDATA System Option (p. 579)</td>
<td>Specifies the source database for the locale information.</td>
</tr>
<tr>
<td></td>
<td>LOGLANGENG System Option (p. 581)</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. 583)</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. 585)</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. 588)</td>
<td>Specifies the user local time zone.</td>
</tr>
<tr>
<td>Files: External Files</td>
<td>BOMFILE System Option (p. 574)</td>
<td>Specifies whether to write the byte-order mark (BOM) prefix on Unicode-encoded external files.</td>
</tr>
<tr>
<td></td>
<td>RSASIOTRANSERROR System Option (p. 586)</td>
<td>Displays a transcoding error when invalid data is read from a remote application.</td>
</tr>
<tr>
<td></td>
<td>VALIDMEMNAME= System Option (p. 589)</td>
<td>Specifies the rules for naming SAS data sets, SAS data views, and item stores.</td>
</tr>
<tr>
<td></td>
<td>VALIDVARNAME= System Option (p. 592)</td>
<td>Specifies the rules for valid SAS variable names that can be created and processed during a SAS session.</td>
</tr>
<tr>
<td>Language Control</td>
<td>LSWLANG System Option (p. 582)</td>
<td>Specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation.</td>
</tr>
<tr>
<td>Sort: Procedure Options</td>
<td>SORTSEQ= System Option (p. 587)</td>
<td>Specifies a language-specific collating sequence for the SORT and SQL procedures to use in the current SAS session.</td>
</tr>
</tbody>
</table>

**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=**
- EXTFILES
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.

DFLANG= System Option

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=
LANGUAGECONTROL

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
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. The value LOCALE was added later to give all customers the same feature.

- 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.

**See Also**

`DTRESET System Option`

---

**ENCODING System Option**

Specifies the default character-set encoding for the SAS session.

Valid in: configuration file, SAS invocation  
Category: Environment Control: Language Control  
PROC OPTIONS GROUP= LANGUAGECONTROL

**Syntax**

```
-ENCODING= encoding-value
```

**Required Argument**

`encoding-value`

- UTF-8  

*Note:* UTF-8, Unicode encoding is the only supported encoding.

**Details**

A character-set encoding is a set of characters that have been mapped to numeric values called code points.

UTF-8 is the only SAS session encoding supported by SAS Viya. The encoding for a SAS session is determined by the values of the ENCODING= option. If ENCODING= is not specified, UTF-8 is set as default.
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

**Operating environment:** Also valid in SASV9_OPTIONS environment variable

**Syntax**

-LOCALE locale-name

LOCALE=locale-name

**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 629.

**Details**

The LOCALE= system option is used to specify the locale, which reflects the local conventions, language, and culture 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.

When you set a value for LOCALE=, the value of the following system options is modified unless explicit values have been specified:

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.

See Also

Conceptual Information:
• Chapter 2, “Locale for NLS,” on page 5
• “LOCALE= Values for PAPERSIZE and DFLANG Options” on page 629

System Options:
• “ENCODING System Option” on page 577
• “DFLANG= System Option” on page 575

LOCALEDATA System Option
Specifies the source database for the locale information.

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS
GROUP= LANGUAGECONTROL
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 text output to the SAS log can be changed.

Valid in: configuration file, SAS invocation

PROC OPTIONS GROUP= LOGCONTROL

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 585
- “LSWLANG System Option” on page 582
- “LOGLANGENG System Option” on page 581
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\nls) 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 582
- “LOGLANGCHG System Option” on page 580
- “ODSLANGCHG System Option” on page 585

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

LSWLANG=LOCALE | language

Required Arguments

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.

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>
</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=** LANGUAGECONTROL
- **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;
```
data b;
  y=&macx
run;

See Also
“BESTDOTX Format” on page 87

ODSLANGCHG System Option
Determined whether the language of the text of the ODS output can be changed

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS GROUP= LANGUAGECONTROL
Default: ODSLANGCHG is set to off in all servers except for the UNICODE server
Tip: The language used for the language switching feature is controlled by LSWLANG= option. By default, LSWLANG is set to LOCALE, which specifies that the language of LSW is controlled by the language of LOCALE.

Syntax
ODSLANGCHG | NOODSLANGCHG

Syntax Description
ODSLANGCHG
   Specifies that the language of ODS output can change after start-up.

NOODSLANGCHG
   Specifies that the language of ODS output cannot change after start-up.

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 581
- “LOGLANGCHG System Option” on page 580
- “LSWLANG System Option” on page 582

**RSASIOTRANSERROR System Option**

Displays a transcoding error when invalid data is read from a remote application.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>configuration file, SAS invocation, OPTIONS statement, SAS System Options window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Files: SAS Files</td>
</tr>
<tr>
<td>PROC OPTIONS GROUP=</td>
<td>SASFILES</td>
</tr>
<tr>
<td>Default:</td>
<td>RSASIOTRANSERROR</td>
</tr>
</tbody>
</table>

**Syntax**

RSASIOTRANSERROR | NORSASIOTRANSERROR
Syntax Description

**RSASIOTRANSERROR**
- specifies to display a transcoding error when invalid values are read from a remote application.

**NORSASIOTRANSERROR**
- specifies not to display a transcoding error when invalid values are read from a remote application.

Details

The RSASIOTRANSERROR 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**

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

**Syntax**

SORTSEQ= | <LINGUISTIC>

**Syntax Description**

**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 *SAS Viya Visual Data Management and Utility Procedures Guide* for more information.

**Example**

This example demonstrates the functionality of SORTSEQ with PROC SORT and PROC SQL:

```sas
options sortseq=linguistic;
proc sort data=sashelp.class out=f001;
   by name;
   run;
proc sql;
   create table f002 as select * from sashelp.class order by name;
   quit;
run;
```
SAS provides ICU collation when the linguistic option (SORTSEQ=LINGUISTIC) is specified on the Base SAS procedure, PROC SORT. Starting in the third maintenance release of SAS 9.4, 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 in-depth information about the UCA algorithm or the International Components for Unicode (ICU) library implementation, see Download the ICU 4.8 Release and CLDR 2.0 Release Note.

See Also

“Collating Sequence” on page 14

---

**TIMEZONE= System Option**

Specifies the user local time zone.

Valid in: Configuration file, SAS command, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Environment Control: Language Control

PROC OPTIONS GROUP=

LanguageControl

Alias: TZ=, except in the restricted options configuration file where TIMEZONE= must be used

Note: This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

**Syntax**

```plaintext
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 2, “Time Zone IDs and Time Zone Names,” on page 663.

*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 2, “Time Zone IDs and Time Zone Names,” on page 663.

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( ), TZONENAME( ), TZONEID( ), TZONEU2S( ), and TZONE2U( ),
• time zone formats B8601DXw, E8601DXw, B8601LXw, E8601LXw, B8601TXw, E8601TXw, NLDATMWZw, NLDATMTZw, and NLDATMZw.

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 Daylight Time, Cuba Daylight Time, and China Daylight 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 25
Default: The shipped default is COMPATIBLE.

Applies to: BASE engine

Restriction: Regardless of the value of VALIDMEMNAME, a member name cannot end in the special character # followed by three digits. This is because it would conflict with the naming conventions for generation data sets. Using such a member name results in an error.

Interaction: SAS Studio sets VALIDMEMNAME=COMPAT before and after each code submission. For more information, see “System Options in SAS Studio” in SAS Viya System Options: Reference.

Note: This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

Syntax

VALIDMEMNAME=COMPATIBLE | EXTEND

Syntax Description

COMPATIBLE specifies that a SAS data set name, a SAS data view name, or an item store name must follow these rules:

- The length of the names can be up to 32 characters.
- 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.
- 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).
- 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.

Operating environment  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'.
```

Tip  The name is displayed in uppercase letters.

Examples  data “August Purchases”;

data ‘Años de empleo’;

CAUTION  Throughout SAS, using the name literal syntax with SAS member names that exceed the 32-byte limit or that have excessive embedded quotation marks might cause unexpected results. The intent of the VALIDMEMNAME=EXTEND system option is to enable compatibility with other DBMS member naming conventions, such as allowing embedded blanks and national characters.

Details  When VALIDMEMNAME= EXTEND, valid characters that are allowed in a SAS data set name, SAS data view name, and an item store name are extended to these characters:
• international characters
• characters supported by third-party databases
• characters that are commonly used in a filename

Only the DATA, VIEW, and ITEMSTOR SAS member types support the extension of characters. The other member types, such as CATALOG and PROGRAM, do not 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 592
VALIDVARNAME= System Option

Specifies the rules for valid SAS variable names that can be created and processed during a SAS session.

Valid in:  Configuration file, SAS command, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Files: SAS Files

PROC OPTIONS
GROUP=SASFILES

Default: The shipped default is V7.

Interaction: SAS Studio sets VALIDVARNAME= to the Preferences setting before each code submission. For more information, see “System Options in SAS Studio” in SAS Viya System Options: Reference.

Note: This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

Syntax

VALIDVARNAME=V7 | UPCASE | ANY

Syntax Description

V7

specifies that variable names must follow these rules:

• The length of a SAS variable names 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, cat, Cat, and 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 contains 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,
cat, Cat, and CAT all represent the same variable.

See

Examples

‘% of profit’n=percent;

‘items@warehouse’n=itemnum;

CAUTION

Throughout SAS, using the name literal syntax with SAS member
names that exceed the 32-byte limit or have excessive embedded
quotation marks might cause unexpected results. The intent of the
VALIDVARNANE=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 589
Part 9

Options for Commands, Statements, and Procedures for NLS

Chapter 16

Dictionary of Command, Statement, and Procedure Option for NLS
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=, and CVPMULTIPLIER= Options (p. 605)</td>
<td>Specifies attributes for character variables that are needed to transcode a SAS file.</td>
</tr>
<tr>
<td></td>
<td>ENCODING= Option (p. 612)</td>
<td>Overrides and transcodes the encoding for input or output processing of external files.</td>
</tr>
<tr>
<td></td>
<td>INENCODING= and OUTENCODING= Options (p. 615)</td>
<td>Overrides and changes the encoding when reading or writing SAS data sets in the SAS library.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ODSCHARSET= Option (p. 617)</td>
<td></td>
<td>Specifies the character set to be generated in the META declaration for the output.</td>
</tr>
<tr>
<td>ODSTRANTAB= Option (p. 618)</td>
<td></td>
<td>Specifies the translation table to use when transcribing an XML document for an output file.</td>
</tr>
<tr>
<td>XMLENCODING= Option (p. 622)</td>
<td></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 (p. 619)</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 transcribed.</td>
</tr>
<tr>
<td>ODS: Third-Party Formatted</td>
<td>CHARSET= Option (p. 598)</td>
<td>Specifies the character set to be generated in the META declaration for the output.</td>
</tr>
<tr>
<td></td>
<td>TRANTAB= Option (p. 621)</td>
<td>Specifies the translation table to use when you are transcoding character data.</td>
</tr>
</tbody>
</table>

**Dictionary**

**CHARSET= Option**

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

    <META http-equiv="Content-Type" content="text/html; charset=utf-8">

See Also

Conceptual Information:

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

Collating Sequence Option

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 option(s)> ;

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.

**DANISH NORWEGIAN**

sorts characters according to the Danish and Norwegian

The Danish and Norwegian collating sequence is shown in Figure 16.1 on page 600.

**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 16.1 on page 600.

**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 14.

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.

The available translation tables are

• DANISH
• EBCDIC
• FINNISH
• ITALIAN
• NORWEGIAN
• POLISH
• REVERSE
• SPANISH
• SWEDISH

The following figure shows how the alphanumeric characters in each language sorts:

Figure 16.1 Alphanumeric Characters Sorted for Each Language

<table>
<thead>
<tr>
<th>Language</th>
<th>Sorted Characters</th>
</tr>
</thead>
</table>
| Danish     | 0123456789
| Finnish    | 0123456789
| Italian    | 0123456789
| Norwegian  | 0123456789
| Spanish    | 0123456789
| Swedish    | 0123456789

Tip
You can specify only one collating-sequence-option in a PROC SORT step.

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:

```plaintext
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 643.

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 643.
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 on the PROC
SORT SORTSEQ= option and PROC SQL.

Notes
Linguistic collation can also be specified using the SORTSEQ= option in the SQL Procedure and by specifying system option
SORTSEQ=LINGUISTIC. For more information, see “SORTSEQ=
System Option” on page 587 and “SORT Procedure” in SAS Viya

Collating rules are supported only for PROC SORT, not for the
system option or SORTSEQ= on PROC SQL.

Tips
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 Viya Visual Data Management and
Utility Procedures Guide

The “Collating Sequence” on page 14 for detailed information
about linguistic collation.

The website for the Unicode Collation Algorithm (UCA)

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER</td>
<td>Sorts uppercase letters first, then the lowercase letters.</td>
</tr>
<tr>
<td>LOWER</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>BIG5HAN</td>
<td>specifies pinyin ordering for Latin and specifies big5 charset ordering for Chinese, Japanese, and Korean characters.</td>
</tr>
<tr>
<td>DIRECT</td>
<td>specifies a Hindi variant.</td>
</tr>
<tr>
<td>GB2312HAN</td>
<td>specifies pinyin ordering for Latin and specifies gb2312han charset ordering for Chinese, Japanese, and Korean characters.</td>
</tr>
<tr>
<td>PHONEBOOK</td>
<td>specifies a telephone-book style for ordering of characters. Select PHONEBOOK only with the German language.</td>
</tr>
<tr>
<td>PINYIN</td>
<td>specifies an ordering for Chinese, Japanese, and Korean characters based on character-by-character transliteration into pinyin. This ordering is typically used with simplified Chinese.</td>
</tr>
<tr>
<td>POSIX</td>
<td>is the Portable Operating System Interface. This option specifies a &quot;C&quot; locale ordering of characters.</td>
</tr>
<tr>
<td>STROKE</td>
<td>specifies a nonalphabetic writing style ordering of characters. Select STROKE with Chinese, Japanese, Korean, or Vietnamese languages. This ordering is typically used with Traditional Chinese.</td>
</tr>
<tr>
<td>TRADITIONAL</td>
<td>specifies a traditional style for ordering of characters. For example, select 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 629 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_UnitedKingdom, 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;ás&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>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;åö&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>
</tbody>
</table>
Table 16.1 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 Danish</td>
<td>DANISH on page 599</td>
</tr>
<tr>
<td>Specify Finnish</td>
<td>FINNISH on page 599</td>
</tr>
<tr>
<td>Specify Norwegian</td>
<td>NORWEGIAN on page 599</td>
</tr>
<tr>
<td>Specify Polish</td>
<td>POLISH on page 599</td>
</tr>
<tr>
<td>Specify Swedish</td>
<td>SWEDISH on page 599</td>
</tr>
<tr>
<td>Specify a customized sequence</td>
<td>NATIONAL on page 599</td>
</tr>
</tbody>
</table>
Task

Specify any of the collating sequences listed above (ASCII, EBCDIC, DANISH, FINNISH, ITALIAN, NORWEGIAN, POLISH, SPANISH, SWEDISH, or NATIONAL), the name of any other system provided translation table (POLISH, SPANISH), and the name of a user-created translation table. You can specify an encoding. You can also specify either the keyword LINGUISTIC or UCA to achieve a locale-appropriate collating sequence.

See Also

- “Collating Sequence” on page 14

System Options:

- “SORTSEQ= System Option” on page 587

CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options

Specifies attributes for character variables that are needed to transcode a SAS file.

Valid in: LIBNAME statement
Category: Data Access
See: LIBNAME, SAS/ACCESS

Syntax

LIBNAME libref'SAS data-library'; <CVPBYTES=bytes> <CVPENGINE=engine> <CVPMULTIPLIER=multiplier> <CVPVARCHAR=YES | NO>

Optional Arguments

CVPBYTES=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 32766.

For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPBYTES= 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.
If you specify CVPBYTES=, SAS automatically uses the CVP engine in order to expand the character variable lengths according to your specification. If you explicitly assign the CVP engine but do not specify either CVPBYTES= or CVPMULTIPLIER=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths of the character variables.

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.

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.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=.

Specify one of these options.

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.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

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.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

You cannot specify both
your specification. If you explicitly specify the CVP engine but do not specify either CVPMULTIPLIER= or CVPBYTES=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths.

**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 in the SAS log.

**Interaction**
You cannot specify both CVPMULTIPLIER= and CVPBYTES=. Specify one of these options.

**See**
“Avoiding Character Data Truncation By Using the CVP Engine” on page 23

**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.

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.
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>ś</td>
<td>l</td>
</tr>
</tbody>
</table>
The following code is executed in SAS Viya. The default session encoding in UTF-8.

```sas
options msglevel=i;
```

### PROC CONTENTS Output for MYFILES.WLATIN2

#### The SAS System

The `CONTENTS` Procedure

<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>08/17/2016 09:00:44</td>
<td>Observation Length</td>
<td>4</td>
</tr>
<tr>
<td>Last Modified</td>
<td>08/17/2016 09:00:44</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_32</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>U:\wlatin2.sas7bdat</td>
</tr>
<tr>
<td>Release Created</td>
<td>9.0401M3</td>
</tr>
<tr>
<td>Host Created</td>
<td>W32_7PRO</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 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.

Log 16.1  SAS Log with Transcoding Error

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 may require additional CPU resources and reduce performance.
ERROR: Some character data was lost during transcoding in the data set MYFILES.UTF8.
NOTE: The data 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 in SAS Viya and expands the variable lengths:

options msglevel=i;
libname myfiles 'SAS data-library';
libname expand cvp 'SAS data-library' cvpbytes=2;
data myfiles.utf8;
   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:
Log 16.2  SAS Log Output for MYFILES.UTF8

57     options msglevel=1;
58          libname myfiles 'directory path';
NOTE: Libref MYFILES was successfully assigned as follows:
      Engine:        V9
      Physical Name: /directory path
59          libname expand cvp '/directory path' 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: /directory path
60         data myfiles.utf8;
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.
NOTE: DATA statement used (Total process time):
      real time          0.02 seconds
      cpu time           0.01 seconds
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:
ENCODING= Option

Overrides and transcodes the encoding for input or output processing of external files.

**Valid in:** %INCLUDE statement; FILE statement; FILENAME 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 643.

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.</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>
<tr>
<td>FILE command:</td>
<td>saves the contents of a window to an external file.</td>
</tr>
<tr>
<td>INCLUDE command:</td>
<td>Copies an external file into the current window.</td>
</tr>
</tbody>
</table>

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 UTF-8, but the external file's encoding needs to be LATIN1. 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:

```
libname myfiles 'SAS data-library';
```
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 WLATIN1 character-set encoding, and the current SAS session is in the UTF-8 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;
run;
```

When you specify that the external file is in WLATIN1, SAS then transcodes the external file from WLATIN1 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 UTF-8.

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 UTF-8, but the external file's encoding needs to be WLATIN1.

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="WLATIN1";
data _null_;  
set myfiles.cars;  
file outfile;  
put Make Model Year;  
run;
```

When you specify that the external file is to be in WLATIN1 encoding, SAS then transcodes the data from UTF-8 to the specified encoding, WLATIN1, 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 WLATIN2, and the current SAS session encoding is UTF-8. 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:

```sas
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 WLATIN2, SAS then transcodes the external file from WLATIN2 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 UTF-8.

**See Also**

**Statements:**

- “FILE Statement” in *SAS Viya Statements: Reference*
- “FILENAME Statement” in *SAS Viya Statements: Reference*
- “INFILE Statement” in *SAS Viya Statements: Reference*

---

**INENCODING= and OUTENCODING= Options**

Overrides and changes the encoding when reading or writing SAS data sets in the SAS library.

**Valid in:** LIBNAME statement

**Category:** Data Access
**Syntax**

\[
\text{INENCODING}= \text{ANY} | \text{ASCIIANY} | \text{EBCDICANY} | \text{encoding-value}
\]

\[
\text{OUTENCODING}= \text{ANY} | \text{ASCIIANY} | \text{EBCDICANY} | \text{encoding-value}
\]

**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 18, “Encoding Values in SAS Language Elements,” on page 643.

**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.

**See Also**

- “Overview: Encoding for NLS” on page 9
ODSCHARSET= Option

Specifies the character set to be generated in the META declaration for the output.

Valid in: LIBNAME statement for the XMLV2 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:


ODSTRANTAB= Option

Specifies the translation table to use when transcoding an XML document for an output file.

Valid in: the LIBNAME statement for the XMLV2 engine

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

The preferred method for specifying an encoding is to use the LOCALE= system option.

See Also

Conceptual Information:

- Conceptual discussion of Chapter 2, “Locale for NLS,” on page 5

System Options:

- “LOCALE System Option” on page 578

Statements:


---

TRANS CODE= 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

- Chapter 4, “Transcoding for NLS,” on page 19

TRANSCODE= Option

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

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 Viya Statements: Reference.

TRANSCODE= YES | NO

Specifies whether to transcode character variables. Use TRANSCODE=NO to suppress transcoding. For more information, see “Overview to Transcoding” on page 19.
The TRANSCODE= attribute is not supported by some SAS Workspace Server clients. Variables with TRANSCODE=NO are not returned in SAS Viya. 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 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;
  length z $4;
  z = 'snow';
  attrib z transcode = YES;
data a;
  set b;
  set c;
  /* Check transcode setting for variable Z */
  rcl = vtranscode(z);
  put rcl=;
run;
```

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.

```sas
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 */
  rcl = vtranscode(z);
  put rcl=;
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 427
- “VTRANSCODEX Function” on page 428

TRANTAB= Option

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

The ENCODING= option is preferred when specifying the encoding.
**XMLENCODING= Option**

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 643.

The default for `encoding-value` is the current session encoding.

**See Also**

**Statements:**


---

**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 | (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

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.

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-option(s)-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.

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 19
Part 10

Values for Locale and Transcoding

Chapter 17
Values for the LOCALE= System Option .......................... 629

Chapter 18
Encoding Values in SAS Language Elements .................. 643
Chapter 17
Values for the LOCALE= System Option

LOCALE= Values for PAPERSIZE and DFLANG Options

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.

Table 17.1 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 af</td>
</tr>
<tr>
<td>Albanian_Albania</td>
<td>sq_AL</td>
<td>Albanian 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>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>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</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>Arabic_Yemen</td>
<td>ar_YE</td>
<td>ar</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>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</td>
</tr>
<tr>
<td>Chinese_HongKong</td>
<td>zh_HK</td>
<td></td>
</tr>
<tr>
<td>Chinese_Macau</td>
<td>zh_MO</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>Chinese_Singapore</td>
<td>zh_SG</td>
<td></td>
</tr>
<tr>
<td>Chinese_Taiwan</td>
<td>zh_TW</td>
<td></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></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></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></td>
</tr>
<tr>
<td>English_Belgium</td>
<td>en_BE</td>
<td></td>
</tr>
<tr>
<td>English_Botswana</td>
<td>en_BW</td>
<td></td>
</tr>
<tr>
<td>English.Canada</td>
<td>en_CA</td>
<td></td>
</tr>
<tr>
<td>English_Caribbean</td>
<td>en_CB</td>
<td></td>
</tr>
<tr>
<td>English_HongKong</td>
<td>en_HK</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>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>English_Singapore</td>
<td>en_SG</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>et</td>
</tr>
<tr>
<td>Faroese_FaroeIslands</td>
<td>fo_FO</td>
<td>Faroese</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fo</td>
</tr>
<tr>
<td>Finnish_Finland</td>
<td>fi_FI</td>
<td>Finnish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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>German_Germany</td>
<td>de_DE</td>
<td>German</td>
</tr>
<tr>
<td></td>
<td></td>
<td>de</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></td>
</tr>
<tr>
<td>Greek_Greece</td>
<td>el_GR</td>
<td>Greek</td>
</tr>
<tr>
<td></td>
<td></td>
<td>el</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>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>Korean_Korea</td>
<td>ko_KR</td>
<td>Korean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ko</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Korean_SouthKorea</td>
</tr>
<tr>
<td>Latvian_Latvia</td>
<td>lv_LV</td>
<td>Latvian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lv</td>
</tr>
<tr>
<td>Lithuanian_Lithuania</td>
<td>lt_LT</td>
<td>Lithuanian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lt</td>
</tr>
<tr>
<td>Macedonian_Macedonia</td>
<td>mk_MK</td>
<td>Macedonian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mk</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>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 pt</td>
</tr>
<tr>
<td>Portuguese_Portugal</td>
<td>pt_PT</td>
<td>Portuguese pt</td>
</tr>
<tr>
<td>Romanian_Romania</td>
<td>ro_RO</td>
<td>Romanian ro</td>
</tr>
<tr>
<td>Russian_Russia</td>
<td>ru_RU</td>
<td>Russian ru</td>
</tr>
<tr>
<td>Russian_Ukraine</td>
<td>ru_UA</td>
<td></td>
</tr>
<tr>
<td>Serbian_BosniaHerzegovina</td>
<td>sr_BA</td>
<td>Serbian sr</td>
</tr>
<tr>
<td>Serbian_Montenegro</td>
<td>sr_ME</td>
<td></td>
</tr>
<tr>
<td>Serbian_Serbia</td>
<td>sr_RS</td>
<td>Serbian sr</td>
</tr>
<tr>
<td>SerbianLatin_BosniaHerzegovina</td>
<td>sh_BA</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>SerbianLatin_Montenegro</td>
<td>sh_ME</td>
<td>SerbianLatin sh</td>
</tr>
<tr>
<td>SerbianLatin_Serbia</td>
<td>sh_RS</td>
<td>SerbianLatin sh</td>
</tr>
<tr>
<td>Slovak_Slovakia</td>
<td>sk_SK</td>
<td>Slovak Slovakian Slovakian_Slovakia sk</td>
</tr>
<tr>
<td>Slovenian_Slovenia</td>
<td>sl_SI</td>
<td>Slovenian sl</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>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>
</tbody>
</table>
The following table lists the valid POSIX values. 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>
<thead>
<tr>
<th>SAS Name</th>
<th>POSIX Locale</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<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>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>
<tr>
<td>Ukrainian_Ukraine</td>
<td>uk_UA</td>
<td>Ukrainian uk</td>
</tr>
<tr>
<td>Vietnamese_Vietnam</td>
<td>vi_VN</td>
<td>Vietnamese vi</td>
</tr>
</tbody>
</table>

Table 17.2  Default Values for the DFLANG, DATESTYLE, and PAPERSIZE System Options Based on the LOCALE= System Option

<table>
<thead>
<tr>
<th>POSIX Locale</th>
<th>DFLANG=</th>
<th>DATESTYLE=</th>
<th>PAPERSIZE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>af_ZA</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ar_AE</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>ar_BH</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_DZ</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_EG</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IQ</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_JO</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_KW</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_LB</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_LY</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_MA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_OM</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_QA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SD</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SY</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_TN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_YE</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>be_BY</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>bg_BG</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>bn_IN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>bs_BA</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ca_ES</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>cs_CZ</td>
<td>Czech</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>da_DK</td>
<td>Danish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_AT</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>de_CH</td>
<td>Swiss_German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_DE</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_LI</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_LU</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>el_CY</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>el_GR</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_AU</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BE</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BW</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_CA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_CB</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>en_GB</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_HK</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IE</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_JM</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_MT</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_NZ</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_PH</td>
<td>English</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>en_SG</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_US</td>
<td>English</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>en_ZA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_ZW</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_AR</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_BO</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>es_CL</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CO</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CR</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_DO</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_EC</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_ES</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_GT</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_HN</td>
<td>Spanish</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>es_MX</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_NI</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PA</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PE</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_PR</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PY</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_SV</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_US</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_UY</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_VE</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>et_EE</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>eu_ES</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IN</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IR</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fi_FI</td>
<td>Finnish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fo_FO</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_BE</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>fr_CA</td>
<td>French</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>fr_CH</td>
<td>Swiss_French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_FR</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_LU</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ga_IE</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>gv_GB</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>he_IL</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>hi_IN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>hr_BA</td>
<td>Croatian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>hr_HR</td>
<td>Croatian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>hu_HU</td>
<td>Hungarian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>id_ID</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>is_IS</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_CH</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_IT</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ja_JP</td>
<td>Japanese</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kk_KZ</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>kl_GL</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ko_KR</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kw_GB</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>lt_LT</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>lv_LV</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>mk_MK</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>mr_IN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ms_MY</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>mt_MT</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nb_NO</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_BE</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_NL</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nn_NO</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>no_NO</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>pl_PL</td>
<td>Polish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>pt_BR</td>
<td>Portuguese</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>pt_PT</td>
<td>Portuguese</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ro_RO</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ru_RU</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ru_UA</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_BA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_ME</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_RS</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sk_SK</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sl_SI</td>
<td>Slovenian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sq_AL</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sr_BA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sr_ME</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sr_RS</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sv_SE</td>
<td>Swedish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ta_IN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>te_IN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>th_TH</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>tr_TR</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>uk_UA</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>vi_VN</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_CN</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_HK</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_MO</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_SG</td>
<td>Locale</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_TW</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
</tbody>
</table>
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 read into SAS Studio connecting to SAS Viya running the UTF-8 encoding. In order for SAS Viya to successfully access the file, the SAS data file must be transcoded from the Western Latin1 encoding to UTF-8 encoding. For information about transcoding concepts, including SAS language elements that contain options for transcoding, see Chapter 4, “Transcoding for NLS,” on page 19.

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>
<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>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>acyrllic</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>ebedic037</td>
<td>e037</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebedic275</td>
<td>e275</td>
<td>Brazil EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebedic424</td>
<td>e424</td>
<td>Hebrew EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebedic425</td>
<td>e425</td>
<td>Arabic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebedic500</td>
<td>e500</td>
<td>International EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebedic838</td>
<td>e838</td>
<td>Thai 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>ebc dic870</td>
<td>e870</td>
<td>Central European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic875</td>
<td>e875</td>
<td>Greek EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic905</td>
<td>e905</td>
<td>Latin 3 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic924</td>
<td>e924</td>
<td>European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1025</td>
<td>ecyr</td>
<td>Cyrillic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1026</td>
<td>etur</td>
<td>Turkish EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1047</td>
<td>elat</td>
<td>Western EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1097</td>
<td>e097</td>
<td>Farsi Bilingual EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1112</td>
<td>ebal</td>
<td>Baltic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1122</td>
<td>eest</td>
<td>Estonian EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1130</td>
<td>evie</td>
<td>Vietnamese EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1137</td>
<td>e137</td>
<td>Devanagari EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1140</td>
<td>e140</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1141</td>
<td>e141</td>
<td>Austria/Germany EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1142</td>
<td>e142</td>
<td>Denmark/Norway EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1143</td>
<td>e143</td>
<td>Finland/Sweden EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1144</td>
<td>e144</td>
<td>Italy EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1145</td>
<td>e145</td>
<td>Spain EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1146</td>
<td>e146</td>
<td>United Kingdom EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1147</td>
<td>e147</td>
<td>France EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1148</td>
<td>e148</td>
<td>International 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>ebc dic1149</td>
<td>e149</td>
<td>Iceland EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1153</td>
<td>e153</td>
<td>Latin 2 Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1154</td>
<td>e154</td>
<td>Cyrillic Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1155</td>
<td>e155</td>
<td>Turkey Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1156</td>
<td>e156</td>
<td>Baltic Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1157</td>
<td>e157</td>
<td>Estonia Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1158</td>
<td>e158</td>
<td>Cyrillic Ukraine Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1160</td>
<td>e160</td>
<td>cp1160 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic1164</td>
<td>e164</td>
<td>cp1164 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc dic any</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>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>jfug</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>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>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>yhpj</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>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>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>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>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>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>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 Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-424</td>
<td>eoiw</td>
<td>Hebrew Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-425</td>
<td>eoa2</td>
<td>Arabic Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-838</td>
<td>eoth</td>
<td>Thai Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-870</td>
<td>eol2</td>
<td>Central European Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-875</td>
<td>eool</td>
<td>Greek Open Edition</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 Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-930</td>
<td>oe30</td>
<td>Katakana Open Edition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-933</td>
<td>oe33</td>
<td>Korean Open Edition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-935</td>
<td>oe35</td>
<td>Simplified Chinese Open Edition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-937</td>
<td>oe37</td>
<td>Traditional Chinese Open Edition</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 Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1026</td>
<td>eotr</td>
<td>Turkish Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1047</td>
<td>eol1</td>
<td>Western Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1097</td>
<td>eofa</td>
<td>Farsi Bilingual Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1112</td>
<td>eobl</td>
<td>Baltic Open Edition</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-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>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 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1153</td>
<td>eo53</td>
<td>Latin 2 Euro OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1154</td>
<td>eo54</td>
<td>Cyrillic Euro OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1155</td>
<td>eo55</td>
<td>Turkey Euro OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1156</td>
<td>eo56</td>
<td>Baltic Euro OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1157</td>
<td>eo57</td>
<td>Estonia Euro OpenEdition 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>open_ed-1158</td>
<td>eo58</td>
<td>Cryrillic Ukraine Euro 1158 OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1160</td>
<td>eo60</td>
<td>1160 OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1164</td>
<td>eo64</td>
<td>1164 OpenEdition EBCDIC</td>
<td>1</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>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>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>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 *</td>
</tr>
<tr>
<td>utf-16le</td>
<td>u16l</td>
<td>Unicode (UTF-16LE)</td>
<td>2 *</td>
</tr>
<tr>
<td>utf-32be</td>
<td>u32b</td>
<td>Unicode (UTF-32BE)</td>
<td>4 **</td>
</tr>
<tr>
<td>utf-32le</td>
<td>u32l</td>
<td>Unicode (UTF-32LE)</td>
<td>4 **</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>wvietnamese</td>
<td>wvie</td>
<td>Vietnamese Windows</td>
<td>1</td>
</tr>
</tbody>
</table>

* UTF-16BE and UTF-16LE have a fixed length of two bytes per character.
** UTF-32BE and UTF-32LE have a fixed length of four bytes per character.
Appendixes

Appendix 1

*Encodings and Their Aliases and Encoding Character Set Compatibility* ........................................ 655

Appendix 2

*Time Zone IDs and Time Zone Names* ........................................ 663
Appendix 1

Encodings and Their Aliases and Encoding Character Set Compatibility

This table lists common encoding methods and their corresponding encoding and alias names.

Table A1.1 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</td>
</tr>
<tr>
<td></td>
<td>WCYRILLIC</td>
<td>CP1251</td>
</tr>
<tr>
<td></td>
<td>WLATIN1</td>
<td>CP1252</td>
</tr>
<tr>
<td></td>
<td>WGREEK</td>
<td>CP1253</td>
</tr>
<tr>
<td></td>
<td>WTURKISH</td>
<td>CP1254</td>
</tr>
<tr>
<td></td>
<td>WHEBREW</td>
<td>CP1255</td>
</tr>
<tr>
<td></td>
<td>WARABIC</td>
<td>CP1256</td>
</tr>
<tr>
<td></td>
<td>WBALTIC</td>
<td>CP1257</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>ISO8859_1, ISO8859-1</td>
</tr>
<tr>
<td></td>
<td>LATIN2</td>
<td>ISO8859_2, ISO8859-2</td>
</tr>
<tr>
<td></td>
<td>LATIN3</td>
<td>ISO8859_3, ISO8859-3</td>
</tr>
<tr>
<td></td>
<td>CYRILLIC</td>
<td>ISO8859_5, ISO8859-5</td>
</tr>
<tr>
<td></td>
<td>ARABIC</td>
<td>ISO8859_6, ISO8859-6</td>
</tr>
<tr>
<td></td>
<td>GREEK</td>
<td>ISO8859_7, ISO8859-7</td>
</tr>
<tr>
<td></td>
<td>HEBREW</td>
<td>ISO8859_8, ISO8859-8</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>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>EBCDIC838, CP838</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-870</td>
<td>EBCDIC870, CP870</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-875</td>
<td>EBCDIC875, CP875</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1047</td>
<td>EBCDIC1047, CP1047</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1122</td>
<td>EBCDIC1122, CP1122</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1130</td>
<td>EBCDIC1130, CP1130</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1141</td>
<td>EBCDIC1141, CP1141</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1142</td>
<td>EBCDIC1142, CP1142</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1143</td>
<td>EBCDIC1143, CP1143</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1144</td>
<td>EBCDIC1144, CP1144</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1145</td>
<td>EBCDIC1145, CP1145</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1146</td>
<td>EBCDIC1146, CP1146</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1147</td>
<td>EBCDIC1147, CP1147</td>
<td></td>
</tr>
<tr>
<td>OPEN_ED-1148</td>
<td>EBCDIC1148, CP1148</td>
<td></td>
</tr>
<tr>
<td>DBCS</td>
<td>SHIFT-JIS</td>
<td>MS-932, IBM-942, MACOS-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS-949, MACOS-3, EUC-KR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EUC-CN, MS-936, MACOS-25, DEC-CN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EUC-TW, DEC-TW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS-950, 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, and OPEN_ED-425. Encoding names used by SAS for the character sets are listed in the fourth column.

Table A1.2  Encoding Character Set Compatibility

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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</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</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, 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</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</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</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</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>---------------------------------------------</td>
<td>-----------------------------------------------</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</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</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</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</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</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</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</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>--------------------------</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</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</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</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</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</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</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</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</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</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>---------------------------------</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</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</td>
</tr>
<tr>
<td>Malay</td>
<td>ms</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN1, PCOEM850, OPEN_ED-1148</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1148</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</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</td>
</tr>
<tr>
<td>Portuguese</td>
<td>pt</td>
<td>LATIN1</td>
<td>WLATIN1, LATIN9, PCOEM850, OPEN_ED-1145</td>
<td>CP1252, ISO8859_1, ISO8859_15, CP850, CP1047</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</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</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</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</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, PCOEM852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870</td>
</tr>
<tr>
<td>Hungarian</td>
<td>hu</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870</td>
</tr>
<tr>
<td>Polish</td>
<td>pl</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870</td>
</tr>
<tr>
<td>Romanian</td>
<td>ro</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870</td>
</tr>
<tr>
<td>Serbian (Latin)</td>
<td>sh</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870</td>
</tr>
<tr>
<td>Slovak</td>
<td>sk</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870</td>
</tr>
<tr>
<td>Slovenian</td>
<td>sl</td>
<td>LATIN2</td>
<td>WLATIN2, LATIN2, PCOEM852, OPEN_ED-870</td>
<td>CP1250, ISO8859_2, ISO8859_16, CP852, CP870</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, PCOEM857, OPEN_ED-1026</td>
<td>CP1254, ISO8859_9, CP857, CP1026</td>
</tr>
<tr>
<td>Estonian</td>
<td>et</td>
<td>LATIN6</td>
<td>WBALTIC, LATIN6, MSDOS775, PCOEM921, OPEN_ED-1112</td>
<td>CP1257, N/A, CP775, CP1112</td>
</tr>
<tr>
<td>Latvian</td>
<td>lv</td>
<td>LATIN6</td>
<td>WBALTIC, LATIN6, MSDOS775, PCOEM921, OPEN_ED-1112</td>
<td>CP1257, N/A, CP775, CP1112</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Lithuanian</td>
<td>lt</td>
<td>LATIN6</td>
<td>WBALTIC, LATIN6, MSDOS775, PCOEM921, OPEN_ED-1112</td>
<td>PCOEM874, THAI</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>WVIETNAMES E, OPEN_ED-1130</td>
<td>CP1258, CP1130</td>
</tr>
</tbody>
</table>
Appendix 2
Time Zone IDs and Time Zone
Names

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/Bamako</td>
<td>GMT</td>
<td>ML</td>
<td>Greenwich Mean Time</td>
<td>00:00</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>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/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>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/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/Ouagadougu</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>ML</td>
<td>Greenwich Mean Time</td>
<td>0.00</td>
</tr>
<tr>
<td>Africa/Casablanca</td>
<td>WET</td>
<td>MA</td>
<td>Western European Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>WEST</td>
<td>MA</td>
<td>Western European Summer Time</td>
<td>01.00</td>
</tr>
<tr>
<td>Africa/El_Aaiun</td>
<td>WET</td>
<td>EH</td>
<td>Western European Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>WEST</td>
<td>EH</td>
<td>Western European Summer Time</td>
<td>01.00</td>
</tr>
<tr>
<td>Africa/Algiers</td>
<td>CET</td>
<td>DZ</td>
<td>Central European Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Algiers</td>
<td>CET</td>
<td>ES</td>
<td>Central European Time</td>
<td>01:00</td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td>ES</td>
<td>Central European Summer Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Tunis</td>
<td>CET</td>
<td>TN</td>
<td>Central European Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Bangui</td>
<td>WAT</td>
<td>CF</td>
<td>West Africa Time</td>
<td>01:00</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/Douala</td>
<td>WAT</td>
<td>CM</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Kinshasa</td>
<td>WAT</td>
<td>CD</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Lagos</td>
<td>WAT</td>
<td>NG</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Libreville</td>
<td>WAT</td>
<td>GA</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Luanda</td>
<td>WAT</td>
<td>AO</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Malabo</td>
<td>WAT</td>
<td>GQ</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Ndjamea</td>
<td>WAT</td>
<td>TD</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Niamey</td>
<td>WAT</td>
<td>NE</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Porto-Novo</td>
<td>WAT</td>
<td>BJ</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Windhoek</td>
<td>WAT</td>
<td>NA</td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td></td>
<td>WAST</td>
<td>NA</td>
<td>West Africa Summer 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>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/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, EET</td>
<td>EG</td>
<td>Eastern European Time</td>
<td>02:00 2:00</td>
</tr>
<tr>
<td>Africa/Tripoli</td>
<td>EET</td>
<td>LY</td>
<td>Eastern European 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>ER</td>
<td>East Africa Time</td>
<td>03:00</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>EAT</td>
<td>SD</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Mogadishu</td>
<td>EAT</td>
<td>SO</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<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>HAST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>America/Adak</td>
<td>HADT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Anchorage</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Anchorage</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Anguilla</td>
<td>AST</td>
<td>AI</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Antigua</td>
<td>AST</td>
<td>AG</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Araguaina</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Buenos Aires</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Catamarca</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/ComodRivadavia</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Cordoba</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Jujuy</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/La_Rioja</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Mendoza</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Rio_Gallegos</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Salta</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/San_Juan</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/San_Luis</td>
<td>ART</td>
<td>AR</td>
<td>Western Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Tucuman</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Ushuaia</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Aruba</td>
<td>AST</td>
<td>AW</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Asuncion</td>
<td>PYT</td>
<td>PY</td>
<td>Paraguay Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Asuncion</td>
<td>PYST</td>
<td>PY</td>
<td>Paraguay Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Atikokan</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Atka</td>
<td>HAST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</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/Atka</td>
<td>HADT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Bahia</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Bahia_Banderas</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Bahia_Banderas</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Barbados</td>
<td>AST</td>
<td>BB</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Belem</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Belize</td>
<td>CST</td>
<td>BZ</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Blanc-Sablon</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Boa_Vista</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Bogota</td>
<td>COT</td>
<td>CO</td>
<td>Colombia Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Boise</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Boise</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Buenos_Aires</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cambridge_Bay</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Cambridge_Bay</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Campo_Grande</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Campo_Grande</td>
<td>AMST</td>
<td>BR</td>
<td>Amazon Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cancun</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Cancun</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Caracas</td>
<td>VET</td>
<td>VE</td>
<td>Venezuela Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>America/Catamarca</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cayenne</td>
<td>GFT</td>
<td>GF</td>
<td>French Guiana Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cayman</td>
<td>EST</td>
<td>KY</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Chicago</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Chicago</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Chihuahua</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Chihuahua</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Coral_Harbour</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</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/Cordoba</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Costa_Rica</td>
<td>CST</td>
<td>CR</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Creston</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Cuiaba</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Cuiaba</td>
<td>AMST</td>
<td>BR</td>
<td>Amazon Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Curacao</td>
<td>AST</td>
<td>CW</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Danmarkshavn</td>
<td>GMT</td>
<td>GL</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>America/Dawson</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Dawson</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Dawson_Creek</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Denver</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Denver</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Detroit</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Detroit</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Dominica</td>
<td>AST</td>
<td>DM</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Edmonton</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Edmonton</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Eirunepe</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/El_Salvador</td>
<td>CST</td>
<td>SV</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Ensenada</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Ensenada</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Fort_Wayne</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Fort_Wayne</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Fortaleza</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Glace_Bay</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Glace_Bay</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Godthab</td>
<td>WGT</td>
<td>GL</td>
<td>West Greenland Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Godthab</td>
<td>WGST</td>
<td>GL</td>
<td>West Greenland Summer Time</td>
<td>-2:00</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/Goose_Bay</td>
<td>AST CA</td>
<td></td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Goose_Bay</td>
<td>ADT CA</td>
<td></td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EST TC</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EDT TC</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Grenada</td>
<td>AST GD</td>
<td></td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Guadeloupe</td>
<td>AST GP</td>
<td></td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Guatemala</td>
<td>CST GT</td>
<td></td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Guayaquil</td>
<td>ECT EC</td>
<td></td>
<td>Ecuador Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Guyana</td>
<td>GYT GY</td>
<td></td>
<td>Guyana Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Halifax</td>
<td>AST CA</td>
<td></td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Halifax</td>
<td>ADT CA</td>
<td></td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Havana</td>
<td>CST CU</td>
<td></td>
<td>Cuba Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Havana</td>
<td>CDT CU</td>
<td></td>
<td>Cuba Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Hermosillo</td>
<td>MST MX</td>
<td></td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EST US</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EDT US</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CST US</td>
<td></td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CDT US</td>
<td></td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EST US</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EDT US</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EST US</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EDT US</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CST US</td>
<td></td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CDT US</td>
<td></td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EST US</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EDT US</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Vincennes</td>
<td>EST US</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Vincennes</td>
<td>EDT US</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</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/Winamac</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Winamac</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indianapolis</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indianapolis</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Inuvik</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Inuvik</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Iqaluit</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Iqaluit</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Jamaica</td>
<td>EST</td>
<td>JM</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Jujuy</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Juneau</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Juneau</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Kentucky/Louisville</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Kentucky/Louisville</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Kentucky/Monticello</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Kentucky/Monticello</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Knox_IN</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Knox_IN</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Kralendijk</td>
<td>AST</td>
<td>BQ</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/La_Paz</td>
<td>BOT</td>
<td>BO</td>
<td>Bolivia Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Lima</td>
<td>PET</td>
<td>PE</td>
<td>Peru Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Los_Angeles</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Los_Angeles</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Louisville</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Louisville</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Lower_Princes</td>
<td>AST</td>
<td>SX</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Maceio</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Managua</td>
<td>CST</td>
<td>NI</td>
<td>Central Standard Time</td>
<td>-6:00</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/Manaus</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Marigot</td>
<td>AST</td>
<td>MF</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Martinique</td>
<td>AST</td>
<td>MQ</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Matamoros</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Matamoros</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Mazatlan</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Mazatlan</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Mendoza</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Menominee</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Menominee</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Merida</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Merida</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Metlakatla</td>
<td>MEST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Mexico_City</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Mexico_City</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Miquelon</td>
<td>PMST</td>
<td>PM</td>
<td>Saint Pierre and Miquelon Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Miquelon</td>
<td>PMDT</td>
<td>PM</td>
<td>Saint Pierre and Miquelon Daylight Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/Moncton</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Moncton</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Monterrey</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Monterrey</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Montevideo</td>
<td>UYT</td>
<td>UY</td>
<td>Uruguay Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Montevideo</td>
<td>UYST</td>
<td>UY</td>
<td>Uruguay Summer Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/Montserrat</td>
<td>AST</td>
<td>MS</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Nassau</td>
<td>EST</td>
<td>BS</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Nassau</td>
<td>EDT</td>
<td>BS</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/New_York</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</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/New_York</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Nipigon</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Nipigon</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Nome</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Nome</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Noronha</td>
<td>FNT</td>
<td>BR</td>
<td>Fernando de Noronha Standard Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/North_Dakota/Beulah</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/North_Dakota/Beulah</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/North_Dakota/Center</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/North_Dakota/Center</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/North_Dakota/New_Salem</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/North_Dakota/New_Salem</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Ojinaga</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Ojinaga</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Panama</td>
<td>EST</td>
<td>PA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Pangnirtung</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Pangnirtung</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Paramaribo</td>
<td>SRT</td>
<td>SR</td>
<td>Suriname Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Phoenix</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Port-au-Prince</td>
<td>EST</td>
<td>HT</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Port-au-Prince</td>
<td>EDT</td>
<td>HT</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Port_of_Spain</td>
<td>AST</td>
<td>TT</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Porto_Acre</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Porto_Velho</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Puerto_Rico</td>
<td>AST</td>
<td>PR</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Rainy_River</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Rainy_River</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</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/Rankin_Inlet</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Rankin_Inlet</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Recife</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Regina</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Resolute</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Resolute</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Rio_Branco</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Rosario</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Santa_Isabel</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Santa_Isabel</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Santarem</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Santiago</td>
<td>CLT</td>
<td>CL</td>
<td>Chile Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Santiago</td>
<td>CLST</td>
<td>CL</td>
<td>Chile Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Santo_Domingo</td>
<td>AST</td>
<td>DO</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Sao_Paulo</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Sao_Paulo</td>
<td>BRST</td>
<td>BR</td>
<td>Brasilia Summer Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/Scoresbysund</td>
<td>EGT</td>
<td>GL</td>
<td>East Greenland Standard Time</td>
<td>-1:00</td>
</tr>
<tr>
<td>America/Scoresbysund</td>
<td>EGST</td>
<td>GL</td>
<td>East Greenland Summer Time</td>
<td>0:00</td>
</tr>
<tr>
<td>America/Shiprock</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Shiprock</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Sitka</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Sitka</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/St_Barthelemy</td>
<td>AST</td>
<td>BL</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/St_Johns</td>
<td>NST</td>
<td>CA</td>
<td>Newfoundland Standard Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>America/St_Johns</td>
<td>NDT</td>
<td>CA</td>
<td>Newfoundland Daylight Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>America/St_Kitts</td>
<td>AST</td>
<td>KN</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/St_Lucia</td>
<td>AST</td>
<td>LC</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/St_Thomas</td>
<td>AST</td>
<td>VI</td>
<td>Atlantic Standard Time</td>
<td>-4:00</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/St_Vincent</td>
<td>AST</td>
<td>VC</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Swift_Current</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Tegucigalpa</td>
<td>CST</td>
<td>HN</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Thule</td>
<td>AST</td>
<td>GL</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Thule</td>
<td>ADT</td>
<td>GL</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Thunder_Bay</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Thunder_Bay</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Tijuana</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Tijuana</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Toronto</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Toronto</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Tortola</td>
<td>AST</td>
<td>VG</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Vancouver</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Vancouver</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Virgin</td>
<td>AST</td>
<td>TT</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Whitehorse</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Whitehorse</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Winnipeg</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Winnipeg</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Yakutat</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Yakutat</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Yellowknife</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Yellowknife</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</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>WST</td>
<td>AQ</td>
<td>Australian Western Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Antarctica/Davis</td>
<td>DAVT</td>
<td>AQ</td>
<td>Davis Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Antarctica/DumontDUrville</td>
<td>DDUT</td>
<td>AQ</td>
<td>Dumont-d’u2019Urville Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Antarctica/Macquarie</td>
<td>MIST</td>
<td>AU</td>
<td>Macquarie Island Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Antarctica/Mawson</td>
<td>MAWT</td>
<td>AQ</td>
<td>Mawson Time</td>
<td>5:00</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>CLT</td>
<td>AQ</td>
<td>Chile Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Antarctica/Palmer</td>
<td>CLST</td>
<td>AQ</td>
<td>Chile Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Antarctica/Rothera</td>
<td>ROTT</td>
<td>AQ</td>
<td>Rothera Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Antarctica/South_Pole</td>
<td>NZST</td>
<td>NZ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Antarctica/South_Pole</td>
<td>NZDT</td>
<td>NZ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Antarctica/Syowa</td>
<td>SYOT</td>
<td>AQ</td>
<td>Syowa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Antarctica/Vostok</td>
<td>VOST</td>
<td>AQ</td>
<td>Vostok Time</td>
<td>6:00</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>AST</td>
<td>YE</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Almaty</td>
<td>ALMT</td>
<td>KZ</td>
<td>East Kazakhstan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Amman</td>
<td>EET</td>
<td>JO</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Amman</td>
<td>EEST</td>
<td>JO</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Anadyr</td>
<td>ANAT</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Asia/Aqtau</td>
<td>AQTT</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>5:00</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/Aqtobe</td>
<td>AQTT</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Ashgabat</td>
<td>TMT</td>
<td>TM</td>
<td>Turkmenistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Ashkhabad</td>
<td>TMT</td>
<td>TM</td>
<td>Turkmenistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Baghdad</td>
<td>AST</td>
<td>IQ</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Bahrain</td>
<td>AST</td>
<td>BH</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Baku</td>
<td>AZT</td>
<td>AZ</td>
<td>Azerbaijan Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Baku</td>
<td>AZST</td>
<td>AZ</td>
<td>Azerbaijan Summer Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Bangkok</td>
<td>ICT</td>
<td>TH</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Beijing</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Beirut</td>
<td>EET</td>
<td>LB</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Beirut</td>
<td>EEST</td>
<td>LB</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Bishkek</td>
<td>KGT</td>
<td>KG</td>
<td>Kyrgyzstan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Brunei</td>
<td>BNT</td>
<td>BN</td>
<td>Brunei Darussalam Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Calcutta</td>
<td>IST</td>
<td>IN</td>
<td>India Standard Time</td>
<td>5:30</td>
</tr>
<tr>
<td>Asia/Choibalsan</td>
<td>CHOT</td>
<td>MN</td>
<td>Choibalsan Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Chongqing</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Chungking</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Colombo</td>
<td>IST</td>
<td>LK</td>
<td>India Standard Time</td>
<td>5:30</td>
</tr>
<tr>
<td>Asia/Dacca</td>
<td>BDT</td>
<td>BD</td>
<td>Bangladesh Standard Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Damascus</td>
<td>EET</td>
<td>SY</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Damascus</td>
<td>EEST</td>
<td>SY</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Dhaka</td>
<td>BDT</td>
<td>BD</td>
<td>Bangladesh Standard Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Dili</td>
<td>TLT</td>
<td>TL</td>
<td>East Timor Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Dubai</td>
<td>GST</td>
<td>AE</td>
<td>Gulf Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Dushanbe</td>
<td>TJT</td>
<td>TJ</td>
<td>Tajikistan Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Gaza</td>
<td>EET</td>
<td>PS</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Gaza</td>
<td>EEST</td>
<td>PS</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Harbin</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</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/Hebron</td>
<td>EET</td>
<td>PS</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Hebron</td>
<td>EEST</td>
<td>PS</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Ho_Chi Minh</td>
<td>ICT</td>
<td>VN</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Hong Kong</td>
<td>HKT</td>
<td>HK</td>
<td>Hong Kong Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Hovd</td>
<td>HOVT</td>
<td>MN</td>
<td>Hovd Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Irkutsk</td>
<td>IRKT</td>
<td>RU</td>
<td>Irkutsk Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Istanbul</td>
<td>EET</td>
<td>TR</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Istanbul</td>
<td>EEST</td>
<td>TR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Jakarta</td>
<td>WIB</td>
<td>ID</td>
<td>Western Indonesia Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Jayapura</td>
<td>WIT</td>
<td>ID</td>
<td>Eastern Indonesia Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Jerusalem</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Jerusalem</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Kabul</td>
<td>AFT</td>
<td>AF</td>
<td>Afghanistan Time</td>
<td>4:30</td>
</tr>
<tr>
<td>Asia/Kamchatka</td>
<td>PETT</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Asia/Karachi</td>
<td>PKT</td>
<td>PK</td>
<td>Pakistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Kashgar</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kathmandu</td>
<td>NPT</td>
<td>NP</td>
<td>Nepal Time</td>
<td>5:45</td>
</tr>
<tr>
<td>Asia/Katmandu</td>
<td>NPT</td>
<td>NP</td>
<td>Nepal Time</td>
<td>5:45</td>
</tr>
<tr>
<td>Asia/Khandyga</td>
<td>YAKT</td>
<td>RU</td>
<td>Yakutsk Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Asia/Kolkata</td>
<td>IST</td>
<td>IN</td>
<td>India Standard Time</td>
<td>5:30</td>
</tr>
<tr>
<td>Asia/Krasnoyarsk</td>
<td>KRAT</td>
<td>RU</td>
<td>Krasnoyarsk Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kuala Lumpur</td>
<td>MYT</td>
<td>MY</td>
<td>Malaysia Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kuching</td>
<td>MYT</td>
<td>MY</td>
<td>Malaysia Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kuwait</td>
<td>AST</td>
<td>KW</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Macao</td>
<td>CST</td>
<td>MO</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Macao</td>
<td>CST</td>
<td>MO</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Magadan</td>
<td>MAGT</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Asia/Makassar</td>
<td>WITA</td>
<td>ID</td>
<td>Central Indonesia Time</td>
<td>8:00</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/Manila</td>
<td>PHT</td>
<td>PH</td>
<td>Philippine Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Muscat</td>
<td>GST</td>
<td>OM</td>
<td>Gulf Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Nicosia</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Nicosia</td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Novokuznetsk</td>
<td>NOVT</td>
<td>RU</td>
<td>Novosibirsk Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Novosibirsk</td>
<td>NOVT</td>
<td>RU</td>
<td>Novosibirsk Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Omsk</td>
<td>OMST</td>
<td>RU</td>
<td>Omsk Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Oral</td>
<td>ORAT</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Osaka</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Phnom_Penh</td>
<td>ICT</td>
<td>KH</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Pontianak</td>
<td>WIB</td>
<td>ID</td>
<td>Western Indonesia Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Pyongyang</td>
<td>KST</td>
<td>KP</td>
<td>Korean Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Qatar</td>
<td>AST</td>
<td>QA</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Qyzylorda</td>
<td>QYZT</td>
<td>KZ</td>
<td>East Kazakhstan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Rangoon</td>
<td>MMT</td>
<td>MM</td>
<td>Myanmar Time</td>
<td>6:30</td>
</tr>
<tr>
<td>Asia/Riyadh</td>
<td>AST</td>
<td>SA</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Saigon</td>
<td>ICT</td>
<td>VN</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Sakhalin</td>
<td>SAKT</td>
<td>RU</td>
<td>Sakhalin Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Samarkand</td>
<td>UZT</td>
<td>UZ</td>
<td>Uzbekistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Sapporo</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Seoul</td>
<td>KST</td>
<td>KR</td>
<td>Korean Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Shanghai</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Singapore</td>
<td>SGT</td>
<td>SG</td>
<td>Singapore Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Taipei</td>
<td>CST</td>
<td>TW</td>
<td>Taipei Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Tashkent</td>
<td>UZT</td>
<td>UZ</td>
<td>Uzbekistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Tbilisi</td>
<td>GET</td>
<td>GE</td>
<td>Georgia Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Tehran</td>
<td>IRST</td>
<td>IR</td>
<td>Iran Standard Time</td>
<td>3:30</td>
</tr>
<tr>
<td>Asia/Tel_Aviv</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>2:00</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/Tel_Aviv</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Thimbu</td>
<td>BTT</td>
<td>BT</td>
<td>Bhutan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Thimphu</td>
<td>BTT</td>
<td>BT</td>
<td>Bhutan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Tokyo</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Ujung_Pandang</td>
<td>WITA</td>
<td>ID</td>
<td>Central Indonesia Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Ulaanbaatar</td>
<td>ULAT</td>
<td>MN</td>
<td>Ulan Bator Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Ulan_Bator</td>
<td>ULAT</td>
<td>MN</td>
<td>Ulan Bator Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Urumqi</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Ust-Nera</td>
<td>VLAT</td>
<td>RU</td>
<td>Vladivostok Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Vientiane</td>
<td>ICT</td>
<td>LA</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Vladivostok</td>
<td>VLAT</td>
<td>RU</td>
<td>Vladivostok Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Yakutsk</td>
<td>YAKT</td>
<td>RU</td>
<td>Yakutsk Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Asia/Yekaterinburg</td>
<td>YEKT</td>
<td>RU</td>
<td>Yekaterinburg Standard Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Yerevan</td>
<td>AMT</td>
<td>AM</td>
<td>Armenia Standard Time</td>
<td>4:00</td>
</tr>
</tbody>
</table>

### Area: Atlantic

<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 Standard Time</td>
<td>-1:00</td>
</tr>
<tr>
<td>Atlantic/Azores</td>
<td>AZOST</td>
<td>PT</td>
<td>Azores Summer Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Bermuda</td>
<td>AST</td>
<td>BM</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Atlantic/Bermuda</td>
<td>ADT</td>
<td>BM</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Atlantic/Canary</td>
<td>WET</td>
<td>ES</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Canary</td>
<td>WEST</td>
<td>ES</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Cape Verde</td>
<td>CVT</td>
<td>CV</td>
<td>Cape Verde Standard Time</td>
<td>-1:00</td>
</tr>
<tr>
<td>Atlantic/Faeroe</td>
<td>WET</td>
<td>FO</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Faeroe</td>
<td>WEST</td>
<td>FO</td>
<td>Western European Summer Time</td>
<td>1:00</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>Atlantic/Faroe</td>
<td>WET</td>
<td>FO</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Faroe</td>
<td>WEST</td>
<td>FO</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CET</td>
<td>NO</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CEST</td>
<td>NO</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Reykjavik</td>
<td>GMT</td>
<td>IS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/South_Georgia</td>
<td>GST</td>
<td>GS</td>
<td>South Georgia Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Atlantic/St_Helena</td>
<td>GMT</td>
<td>SH</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Stanley</td>
<td>FKST</td>
<td>FK</td>
<td>Falkland Islands Standard Time</td>
<td>-3:00</td>
</tr>
</tbody>
</table>

**Area: Australia**

<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>Australia/ACT</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/ACT</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Brisbane</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Darwin</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</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/Eucla</td>
<td>CWST</td>
<td>AU</td>
<td>Australian Central Western Standard Time</td>
<td>8:45</td>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>LHST</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>LHDT</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Lindeman</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>LHST</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>LHDT</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/North</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Perth</td>
<td>WST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Australia/Queensland</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/South</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/South</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/West</td>
<td>WST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
</tbody>
</table>
## Area: Miscellaneous

<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>1:00</td>
</tr>
<tr>
<td>Arctic/Longyearbyen</td>
<td>CEST</td>
<td>SJ</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Brazil/Acre</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Brazil/DeNoronha</td>
<td>FNT</td>
<td>BR</td>
<td>Fernando de Noronha Standard Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Brazil/East</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Brazil/East</td>
<td>BRST</td>
<td>BR</td>
<td>Brasilia Summer Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Brazil/West</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>CST6CDT</td>
<td>CST</td>
<td></td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>CST6CDT</td>
<td>CDT</td>
<td></td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Canada/Atlantic</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Canada/Atlantic</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Canada/Central</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Central</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Canada/East-Saskatchewan</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Eastern</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Canada/Eastern</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Canada/Mountain</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Canada/Mountain</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Newfoundland</td>
<td>NST</td>
<td>CA</td>
<td>Newfoundland Standard Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>Canada/Newfoundland</td>
<td>NDT</td>
<td>CA</td>
<td>Newfoundland Daylight Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>Canada/Pacific</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>Canada/Pacific</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Canada/Saskatchewan</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Yukon</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>Canada/Yukon</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</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>Chile/Continental</td>
<td>CLT</td>
<td>CL</td>
<td>Chile Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Chile/Continental</td>
<td>CLST</td>
<td>CL</td>
<td>Chile Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Chile/EasterIsland</td>
<td>EAST</td>
<td>CL</td>
<td>Easter Island Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Chile/EasterIsland</td>
<td>EASST</td>
<td>CL</td>
<td>Easter Island Summer Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Cuba</td>
<td>CST</td>
<td>CU</td>
<td>Cuba Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Cuba</td>
<td>CDT</td>
<td>CU</td>
<td>Cuba Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>EST5EDT</td>
<td>EST</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>EST5EDT</td>
<td>EDT</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Egypt</td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td></td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Summer Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Eire</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Eire</td>
<td>IST</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Etc/GMT</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/GMT+0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/GMT+1</td>
<td>GMT-01</td>
<td></td>
<td>1 hours behind GMT</td>
<td>-1:00</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+2</td>
<td>GMT-02</td>
<td></td>
<td>2 hours behind GMT</td>
<td>-2:00</td>
</tr>
<tr>
<td>Etc/GMT+3</td>
<td>GMT-03</td>
<td></td>
<td>3 hours behind GMT</td>
<td>-3:00</td>
</tr>
<tr>
<td>Etc/GMT+4</td>
<td>GMT-04</td>
<td></td>
<td>4 hours behind GMT</td>
<td>-4:00</td>
</tr>
<tr>
<td>Etc/GMT+5</td>
<td>GMT-05</td>
<td></td>
<td>5 hours behind GMT</td>
<td>-5:00</td>
</tr>
<tr>
<td>Etc/GMT+6</td>
<td>GMT-06</td>
<td></td>
<td>6 hours behind GMT</td>
<td>-6:00</td>
</tr>
<tr>
<td>Etc/GMT+7</td>
<td>GMT-07</td>
<td></td>
<td>7 hours behind GMT</td>
<td>-7:00</td>
</tr>
<tr>
<td>Etc/GMT+8</td>
<td>GMT-08</td>
<td></td>
<td>8 hours behind GMT</td>
<td>-8:00</td>
</tr>
<tr>
<td>Etc/GMT+9</td>
<td>GMT-09</td>
<td></td>
<td>9 hours behind GMT</td>
<td>-9:00</td>
</tr>
<tr>
<td>Etc/GMT-0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/GMT-1</td>
<td>GMT+01</td>
<td></td>
<td>1 hours ahead of GMT</td>
<td>1:00</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-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/GMT-2</td>
<td>GMT+02</td>
<td></td>
<td>2 hours ahead of GMT</td>
<td>2:00</td>
</tr>
<tr>
<td>Etc/GMT-3</td>
<td>GMT+03</td>
<td></td>
<td>3 hours ahead of GMT</td>
<td>3:00</td>
</tr>
<tr>
<td>Etc/GMT-4</td>
<td>GMT+04</td>
<td></td>
<td>4 hours ahead of GMT</td>
<td>4:00</td>
</tr>
<tr>
<td>Etc/GMT-5</td>
<td>GMT+05</td>
<td></td>
<td>5 hours ahead of GMT</td>
<td>5:00</td>
</tr>
<tr>
<td>Etc/GMT-6</td>
<td>GMT+06</td>
<td></td>
<td>6 hours ahead of GMT</td>
<td>6:00</td>
</tr>
<tr>
<td>Etc/GMT-7</td>
<td>GMT+07</td>
<td></td>
<td>7 hours ahead of GMT</td>
<td>7:00</td>
</tr>
<tr>
<td>Etc/GMT-8</td>
<td>GMT+08</td>
<td></td>
<td>8 hours ahead of GMT</td>
<td>8:00</td>
</tr>
<tr>
<td>Etc/GMT-9</td>
<td>GMT+09</td>
<td></td>
<td>9 hours ahead of GMT</td>
<td>9:00</td>
</tr>
<tr>
<td>Etc/GMT0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/Greenwich</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/UCT</td>
<td>UCT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/UTC</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/Universal</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/Zulu</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GB</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GB</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>1:00</td>
</tr>
<tr>
<td>GB-Eire</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GB-Eire</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>1:00</td>
</tr>
<tr>
<td>GMT</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GMT+0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GMT-0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GMT0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Greenwich</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------</td>
<td>-------------------------------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>Hongkong</td>
<td>HK</td>
<td>Hong Kong Standard Time</td>
<td>8:00</td>
<td></td>
</tr>
<tr>
<td>Iceland</td>
<td>IS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Antananarivo</td>
<td>MG</td>
<td>East Africa Time</td>
<td>3:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Chagos</td>
<td>IO</td>
<td>Indian Ocean Time</td>
<td>6:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Christmas</td>
<td>CX</td>
<td>Christmas Island Time</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Cocos</td>
<td>CC</td>
<td>Cocos Islands Time</td>
<td>6:30</td>
<td></td>
</tr>
<tr>
<td>Indian/Comoro</td>
<td>KM</td>
<td>East Africa Time</td>
<td>3:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Kerguelen</td>
<td>TF</td>
<td>French Southern and Antarctic Time</td>
<td>5:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Mahe</td>
<td>SC</td>
<td>Seychelles Time</td>
<td>4:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Maldives</td>
<td>MV</td>
<td>Maldives Time</td>
<td>5:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Mauritius</td>
<td>MU</td>
<td>Mauritius Standard Time</td>
<td>4:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Mayotte</td>
<td>YT</td>
<td>East Africa Time</td>
<td>3:00</td>
<td></td>
</tr>
<tr>
<td>Indian/Reunion</td>
<td>RE</td>
<td>Reunion Time</td>
<td>4:00</td>
<td></td>
</tr>
<tr>
<td>Iran</td>
<td>IR</td>
<td>Iran Standard Time</td>
<td>3:30</td>
<td></td>
</tr>
<tr>
<td>Israel</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>2:00</td>
<td></td>
</tr>
<tr>
<td>Israel</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>3:00</td>
<td></td>
</tr>
<tr>
<td>Jamaica</td>
<td>JM</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
<td></td>
</tr>
<tr>
<td>Kwajalein</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
<td></td>
</tr>
<tr>
<td>Libya</td>
<td>LY</td>
<td>Central European Standard Time</td>
<td>2:00</td>
<td></td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MST</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
<td></td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MDT</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
<td></td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
<td></td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
<td></td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
<td></td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
<td></td>
</tr>
<tr>
<td>Mexico/General</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
<td></td>
</tr>
<tr>
<td>Mexico/General</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
<td></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>NZ</td>
<td>NZST</td>
<td>NZ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>NZ</td>
<td>NZDT</td>
<td>NZ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>NZ-CHAT</td>
<td>CHAST</td>
<td>NZ</td>
<td>Chatham Standard Time</td>
<td>12:45</td>
</tr>
<tr>
<td>NZ-CHAT</td>
<td>CHADT</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>-7:00</td>
</tr>
<tr>
<td>Navajo</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>PRC</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>PST8PDT</td>
<td>PST</td>
<td>PST</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>PST8PDT</td>
<td>PDT</td>
<td>PDT</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Poland</td>
<td>CET</td>
<td>PL</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Poland</td>
<td>CEST</td>
<td>PL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Portugal</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Portugal</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>ROC</td>
<td>CST</td>
<td>TW</td>
<td>Taipei Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>ROK</td>
<td>KST</td>
<td>KR</td>
<td>Korean Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Singapore</td>
<td>SGT</td>
<td>SG</td>
<td>Singapore Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Turkey</td>
<td>EET</td>
<td>TR</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Turkey</td>
<td>EEST</td>
<td>TR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>UCT</td>
<td>UCT</td>
<td>UCT</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>US/Alaska</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>US/Alaska</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>US/Aleutian</td>
<td>HAST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>US/Aleutian</td>
<td>HADT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>US/Arizona</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Central</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>US/Central</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/East-Indiana</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/East-Indiana</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</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>US/Eastern</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/Eastern</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>US/Hawaii</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Samoa</td>
<td>SST</td>
<td>AS</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>UTC</td>
<td>UTC</td>
<td>AS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Universal</td>
<td>UTC</td>
<td>AS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>W-SU</td>
<td>MSK</td>
<td>RU</td>
<td>Moscow Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Zulu</td>
<td>UTC</td>
<td>AS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
</tbody>
</table>

**Area: Europe**

<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>Europe/Amsterdam</td>
<td>CET</td>
<td>NL</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Amsterdam</td>
<td>CEST</td>
<td>NL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CET</td>
<td>AD</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CEST</td>
<td>AD</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Athens</td>
<td>EET</td>
<td>GR</td>
<td>Eastern European Standard Time</td>
<td>2:00</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>Europe/Athens</td>
<td>EEST</td>
<td>GR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>BST</td>
<td>GB</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CET</td>
<td>RS</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CEST</td>
<td>RS</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CET</td>
<td>SK</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CEST</td>
<td>SK</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CET</td>
<td>BE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CEST</td>
<td>BE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EET</td>
<td>RO</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EEST</td>
<td>RO</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CET</td>
<td>HU</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CEST</td>
<td>HU</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CET</td>
<td>DK</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CEST</td>
<td>DK</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>IST</td>
<td>IE</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CET</td>
<td>GI</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CEST</td>
<td>GI</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>GMT</td>
<td>GG</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>BST</td>
<td>GG</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Helsinki</td>
<td>EET</td>
<td>FI</td>
<td>Eastern European Standard Time</td>
<td>2:00</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>Europe/Helsinki</td>
<td>EEST</td>
<td>FI</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>GMT</td>
<td>IM</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>BST</td>
<td>IM</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Istanbul</td>
<td>EET</td>
<td>TR</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Istanbul</td>
<td>EEST</td>
<td>TR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>GMT</td>
<td>JE</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>BST</td>
<td>JE</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Kaliningrad</td>
<td>FET</td>
<td>RU</td>
<td>Eastern European Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CET</td>
<td>SI</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CEST</td>
<td>SI</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/London</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/London</td>
<td>BST</td>
<td>GB</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CET</td>
<td>LU</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CEST</td>
<td>LU</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CET</td>
<td>ES</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CEST</td>
<td>ES</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CET</td>
<td>MT</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CEST</td>
<td>MT</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EET</td>
<td>AX</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EEST</td>
<td>AX</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Minsk</td>
<td>FET</td>
<td>BY</td>
<td>Eastern European Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CET</td>
<td>MC</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CEST</td>
<td>MC</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Moscow</td>
<td>MSK</td>
<td>RU</td>
<td>Moscow Standard Time</td>
<td>4:00</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>Europe/Nicosia</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Nicosia</td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Oslo</td>
<td>CET</td>
<td>NO</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Oslo</td>
<td>CEST</td>
<td>NO</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Paris</td>
<td>CET</td>
<td>FR</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Paris</td>
<td>CEST</td>
<td>FR</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Podgorica</td>
<td>CET</td>
<td>ME</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Podgorica</td>
<td>CEST</td>
<td>ME</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Prague</td>
<td>CET</td>
<td>CZ</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Prague</td>
<td>CEST</td>
<td>CZ</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Riga</td>
<td>EET</td>
<td>LV</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Riga</td>
<td>EEST</td>
<td>LV</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Rome</td>
<td>CET</td>
<td>IT</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Rome</td>
<td>CEST</td>
<td>IT</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Samara</td>
<td>SAMT</td>
<td>RU</td>
<td>Moscow Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Europe/San_Marino</td>
<td>CET</td>
<td>SM</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/San_Marino</td>
<td>CEST</td>
<td>SM</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CET</td>
<td>BA</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CEST</td>
<td>BA</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Simferopol</td>
<td>MSK</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CET</td>
<td>MK</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CEST</td>
<td>MK</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EET</td>
<td>BG</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EEST</td>
<td>BG</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CET</td>
<td>SE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CEST</td>
<td>SE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EET</td>
<td>EE</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EEST</td>
<td>EE</td>
<td>Eastern European Summer Time</td>
<td>3:00</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>Europe/Tirane</td>
<td>CET</td>
<td>AL</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Tirane</td>
<td>CEST</td>
<td>AL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CET</td>
<td>LI</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CEST</td>
<td>LI</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CET</td>
<td>VA</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CEST</td>
<td>VA</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CET</td>
<td>AT</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CEST</td>
<td>AT</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EET</td>
<td>LT</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EEST</td>
<td>LT</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Volgograd</td>
<td>VOLT</td>
<td>RU</td>
<td>Volgograd Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CET</td>
<td>PL</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CEST</td>
<td>PL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CET</td>
<td>HR</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CEST</td>
<td>HR</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CET</td>
<td>CH</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CEST</td>
<td>CH</td>
<td>Central European Summer Time</td>
<td>2:00</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>WST</td>
<td>WS</td>
<td>Samoa Standard Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Apia</td>
<td>WSDT</td>
<td>WS</td>
<td>Samoa 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/Chatham</td>
<td>CHAST</td>
<td>NZ</td>
<td>Chatham Standard Time</td>
<td>12:45</td>
</tr>
<tr>
<td>Pacific/Chatham</td>
<td>CHADT</td>
<td>NZ</td>
<td>Chatham Daylight Time</td>
<td>13:45</td>
</tr>
<tr>
<td>Pacific/Chuuk</td>
<td>CHUT</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Easter</td>
<td>EAST</td>
<td>CL</td>
<td>Easter Island Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Pacific/Easter</td>
<td>EASST</td>
<td>CL</td>
<td>Easter Island Summer Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Pacific/Efathe</td>
<td>VUT</td>
<td>VU</td>
<td>Vanuatu Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Enderbury</td>
<td>PHOT</td>
<td>KI</td>
<td>Phoenix Islands Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Fakaofo</td>
<td>TKT</td>
<td>TK</td>
<td>Tokelau Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Fiji</td>
<td>FJT</td>
<td>FJ</td>
<td>Fiji Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Fiji</td>
<td>FJST</td>
<td>FJ</td>
<td>Fiji Summer Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Funafuti</td>
<td>TVT</td>
<td>TV</td>
<td>Tuvalu Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Galapagos</td>
<td>GALT</td>
<td>EC</td>
<td>Galapagos Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Pacific/Gambier</td>
<td>GAMT</td>
<td>PF</td>
<td>Gambier Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>Pacific/Guadalcanal</td>
<td>SBT</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>UM</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Kirimiti</td>
<td>LINT</td>
<td>KI</td>
<td>Line Islands Time</td>
<td>14:00</td>
</tr>
<tr>
<td>Pacific/Kosrae</td>
<td>KOST</td>
<td>FM</td>
<td>Kosrae Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Kwajalein</td>
<td>MHT</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Majuro</td>
<td>MHT</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</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/Marquesas</td>
<td>MART</td>
<td>PF</td>
<td>Marquesas Time</td>
<td>-6:27</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>NRT</td>
<td>NR</td>
<td>Nauru Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Niue</td>
<td>NUT</td>
<td>NU</td>
<td>Niue Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Norfolk</td>
<td>NFT</td>
<td>NF</td>
<td>Norfolk Island Time</td>
<td>11:30</td>
</tr>
<tr>
<td>Pacific/Noumea</td>
<td>NCT</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>PWT</td>
<td>PW</td>
<td>Palau Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Pacific/Pitcairn</td>
<td>PST</td>
<td>PN</td>
<td>Pitcairn Time</td>
<td>-8:00</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>PONT</td>
<td>FM</td>
<td>Ponape Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Port_Moresby</td>
<td>PGT</td>
<td>PG</td>
<td>Papua New Guinea Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Rarotonga</td>
<td>CKT</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>AS</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Tahiti</td>
<td>TAHT</td>
<td>PF</td>
<td>Tahiti Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Tarawa</td>
<td>GILT</td>
<td>KI</td>
<td>Gilbert Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Tongatapu</td>
<td>TOT</td>
<td>TO</td>
<td>Tonga Standard Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Truk</td>
<td>CHUT</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Wake</td>
<td>WAKT</td>
<td>UM</td>
<td>Wake Island Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Wallis</td>
<td>WFT</td>
<td>WF</td>
<td>Wallis and Futuna Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Yap</td>
<td>CHUT</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
</tbody>
</table>
Recommended Reading

Here is the recommended reading list for this title:

- *SAS Viya Data Set Options: Reference*
- *SAS Viya Formats and Informats: Reference*
- *SAS Viya Functions and CALL Routines: Reference*
- *SAS Viya System Options: Reference*
- *SAS Viya Statements: Reference*
- *SAS Encoding - Understanding the Details*

For a complete list of SAS publications, go to [sas.com/store/books](http://sas.com/store/books). If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: [sas.com/store/books](http://sas.com/store/books)
accented character
a type of character that is modified by the addition of an accent mark that alters the pronunciation of the character. An example is "ñ", which results from combining the tilde (~) with the character "n".

American National Standards Institute (ANSI)
the organization that coordinates the development of voluntary consensus standards for products, services, processes, systems, and personnel in the United States. ANSI works with the International Organization for Standardization to establish global standards.

American Standard Code for Information Interchange
See ASCII.

ANSI
See American National Standards Institute.

ASCII (American Standard Code for Information Interchange)
a 7-bit encoding standard that provides a basic set of 128 characters, supporting a variety of computer systems. ASCII encodes the uppercase and lowercase letters of the English alphabet, punctuation marks, the digits 0-9, and control characters. This set of 128 characters is also included in most other encodings. See also EBCDIC, encoding.

bidirectional text
text in a writing system such as Arabic and Hebrew that generally runs from right to left, except for numbers and embedded text written in other languages that run from left to right.

BOM
See byte-order mark.

byte-order mark (BOM)
the Unicode character that indicates the byte order of the Unicode text that follows in the text file or stream. The BOM can also indicate which of several Unicode encodings the text stream is encoded as. In UTF-16, the code point of the byte-order mark is U+FEFF (hexadecimal).

CCS
See coded character set.
CEDA
See Cross-Environment Data Access.

character
the smallest component of a writing system that has a semantic value, such as the letters of an alphabet, digits, or ideographs.

character encoding
a mapping of an abstract character repertoire to a set of numeric values. Character encodings are used in computation, data storage, and transmission of textual data. A character encoding includes national characters, special characters, the digits 0-9, and control characters.

CJK
the languages of Chinese, Japanese, and Korean, referred to as a collective grouping because all use Chinese characters. The grouping is relevant to translation and information processing. Vietnamese is occasionally included in this grouping (CJKV) because of its use of Chinese characters historically.

code page
the representation of a character set that associates a hexadecimal value with each character. The term code page originated from IBMs EBCDIC-based mainframe systems, but many vendors use this term including Microsoft, SAP, and Oracle Corporation. Vendors often allocate their own code page number to a character encoding, even if it is better known by another name (for example, UTF-8 character encoding has code page numbers 1208 at IBM, 65001 at Microsoft, 4110 at SAP). See also encoding, code page.

coded character set (CCS)
a mapping of an abstract character repertoire to a set of numeric values. The ISO Latin-1 coded character set provides the Western European alphabet and symbols and their numeric representations. For example, the letter "Å" is represented as C4 (hexadecimal).

collating sequence
a set of rules that determine how textual data is ordered and compared.

control character
a nonprinting character that is represented by a code point in a character set, and that does not itself represent a written symbol.

Cross-Environment Data Access (CEDA)
a feature of SAS software that enables a SAS data file that was created in a directory-based operating environment to be read by a SAS session in another directory-based environment. See also data representation.

data representation
the form in which data is stored in a particular operating environment. Different operating environments use different standards or conventions for storing floating-point numbers (for example, IEEE or IBM 390); for character encoding (ASCII or EBCDIC); for the ordering of bytes in memory (big Endian or little Endian); for word alignment (4-byte boundaries or 8-byte boundaries); and for data-type length (16-bit, 32-bit, or 64-bit).

DBCS
See double-byte character set.
double-byte character set (DBCS)
a character set that requires a variable-width encoding because many characters
occupy two bytes of memory. The term DBCS, as traditionally applied to languages
such as Japanese, Korean, and Chinese, is somewhat misleading because some
DBCS characters actually require only one byte. See also single-byte character set,
multi-byte character set.

EBCDIC (Extended Binary Coded Decimal Interchange Code)
a family of single-byte and multi-byte encodings for the representation of data on
IBM mainframe and mid-range computers. See also ASCII, encoding.

encode
to represent data in a particular character encoding scheme. For example, in ASCII,
the letter "A" is represented as 41 (hexadecimal).

encoding
a mapping of a coded character set to code values.

encoding method
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.

EUC
See Extended UNIX Code.

Extended Binary Coded Decimal Interchange Code
See EBCDIC.

Extended UNIX Code (EUC)
a multibyte encoding scheme used primarily to encode Chinese, Japanese, and
Korean writing systems with a maximum of 94 characters in sequences of 7-bit
codes.

graphic character
in ISO standards, an encoded character that is intended to be written, printed, or
otherwise displayed in a form that can be read by humans. Graphic characters are
associated with one or more glyphs.

Hangul
the name for the Korean alphabet, consisting of 24 consonant and vowel letters,
arranged into blocks, one per syllable.

I18N
See internationalization.

ICU (International Components for Unicode)
the open-source project containing C/C++ and Java libraries that provide Unicode
and globalization support for software applications.

input method
a type of character conversion that is designed for interactive data input, generally
from a keyboard. The simplest input method works by mapping ASCII characters
onto another alphabet. Another method is to use composition, converting sequences
of characters into a single letter. Languages like Thai and Korean can use both.
International Components for Unicode

See ICU.

International Organization for Standardization (ISO)

an organization that promotes the development of standards, and sponsors related activities that foster the sharing of products, services, and information among nations.

internationalization (I18N)

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. See also national language support.

ISO

See International Organization for Standardization.

ISO 646 family

a group of 7-bit encodings that are defined in the ISO 646 standard. The encoding contains both the 116 invariant ASCII code positions and the 12 variant code positions that can be replaced by national characters. For example, code position 23 (hexadecimal) is reserved for a variant character. This position is occupied by the number sign (#) in the US, and the symbol (£) in the UK. The national variants of ISO 646 are largely obsolete.

ISO 8859 family

a group of 16 8-bit encodings that are defined in the ISO 8859 standard. Each encoding contains both the 128 ASCII characters and the 128 extended characters, which are used in the language or languages that are supported by the encoding. For example, ISO 8859-1, also called Latin-1, is a commonly used encoding in the ISO 8859 family that contains the ASCII characters as well as characters used by Western European languages. See also International Organization for Standardization.

L10N

See localization.

locale

a setting that reflects the language, local conventions, and culture for a geographic region. Local conventions can include specific formatting rules for paper sizes, dates, times, and numbers, and a currency symbol for the country or region. Some examples of locale values are French_Canada, Portuguese_Brazil, and Chinese_Singapore.

localization (L10N)

the process of adapting software for a particular geocultural region (locale). Translation of the user interface, system messages, and documentation is a large part of the localization process. See also internationalization.

logogram

a visual symbol that represents a word or morpheme rather than a speech sound. An example of a logogram in the Chinese language is 山 for the word "mountain".

MBCS

See multi-byte character set.
multi-byte character set (MBCS)
a character set that requires a variable-width encoding because many characters occupy more than one byte of memory. DBCS and MBCS are sometimes used interchangeably, but MBCS is more accurate for describing the character sets of languages such as Japanese, Korean, and Chinese. See also single-byte character set, double-byte character set.

national character
a character that is specific to a language as it is written in a nation or group of nations. For example, the letter “ñ” with a tilde (ñ) is a Spanish national character.

national language support (NLS)
the set of features that enable a software product to function properly in every global market for which the product is targeted.

NLS
See national language support.

SBCS
See single-byte character set.

single-byte character set (SBCS)
a type of encoding for which each character is represented using one byte of computer memory. An example of a single-byte character set is Latin 1. See also double-byte character set, multi-byte character set.

special character
a type of character other than alphanumeric characters, the underscore (_), and the blank. An example is the asterisk (*).

transcoding
the process of converting the contents of a SAS file from one encoding to another encoding. Transcoding is necessary if the session encoding and the file encoding are different, such as when transferring data from a Latin 1 encoding under UNIX to a German EBCDIC encoding on an IBM mainframe. See also encoding, translation table.

translation table
a SAS catalog entry that is used to map data from one encoding to another encoding. SAS language elements that control locale values and encoding properties automatically invoke the appropriate translation table. Translation tables are specific to the operating environment. An example is a translation table that maps the Windows Latin 2 encoding to the ISO Latin 2 encoding. See also encoding, transcoding.

Unicode
a 16-bit encoding that is the industry standard for supporting the interchange, processing, and display of characters and symbols from most of the world's writing systems.
Index

**Special Characters**
- SCPTDW format 88
- SCPTWD format 89
- SEBCDICw. format 90
- SUCS2B format 243
- SUCS2B informat 538
- SUCS2BE format 244
- SUCS2BE informat 539
- SUCS2L format 245
- SUCS2L informat 540
- SUCS2LE format 247
- SUCS2LE informat 542
- SUCS2X format 248
- SUCS2X informat 543
- SUCS2XE format 249
- SUCS2XE informat 544
- SUCS4B format 250
- SUCS4B informat 545
- SUCS4BE format 251
- SUCS4L format 252
- SUCS4L informat 546
- SUCS4LE format 254
- SUCS4X format 255
- SUCS4X informat 547
- SUCS4XE format 256
- SUCS4XE informat 548
- SUESC format 257
- SUESC informat 549
- SUESCE format 259
- SUESCE informat 550
- $UNCR format 260
- $UNCR informat 551
- $UNCRE format 261
- $UNCRE informat 552
- SUPAREN format 262
- SUPAREN informat 553
- SUPARENE format 263
- SUPARENE informat 555
- SUPARENP informat 556
- $UTF8X format 264
- $UTF8X informat 557
- $UTF8XE format 265
- %KCMPRES 562
- %KINDEX Macro Function 562
- %KLEFT macro function 563
- %KLENGTH macro function 563
- %KLOWCASE autocall macro 35
- %KSCAN macro function 564
- %KSUBSTR macro function 566
- %KTRIM autocall macro 36
- %KVERIFY autocall macro 36
- %QCMPRES 562
- %KTRIM autocall macro 36

**Numbers**
- 8859 ISO family 11

**A**
- alignment character expressions 350, 362
- ANORM420 Function 310
- ANSI (American National Standards Institute) 13
- arguments converting to lowercase 354
- converting to uppercase 380
- extracting a substring from 372
- extracting a substring from, based on byte position 375
- length of 353
- transcoding for specified argument 428
- ASCII 11
- transferring data between EBCDIC and 22
- ATTRIB statement 619
- TRANSCODE= option 620
- Australia monetary format 153, 190
- Austria monetary format 200

**B**
- B8601DXw. format 82
- B8601LXw. format 84
- B8601TXw. format 85
BASECHAR function 314
Belgium
monetary format 163, 200
BESTDOTX format 87
binary collation 14
blanks
removing leading DBCS blanks 350
trimming trailing 36
trimming trailing DBCS blanks 362
BOMFILE system option 574
Brazil
monetary format 155, 192
Bulgaria
monetary format 154, 191
Byte Order Mark (BOM) prefix
on Unicode external files 574

C
Call KSCANX function 316
Canada
monetary format 156, 193
case
changing uppercase characters to
lowercase 35
character data
converting to EBCDIC 90
character expressions
comparing 326
compressing 327
concatenating 369
deleting character value contents 381
deleting character value contents, based
on byte unit 383
inserting character value contents 381
inserting character value contents, based
on byte unit 383
left-aligning 350
number of double-byte characters in
328
position of first unique character 386
removing trailing blanks and SO/SI 377
replacing character value contents 381
replacing character value contents, based
on byte unit 383
replacing specific characters 376
reversing 361
right-aligning 362
searching for specific characters 348
searching for string of characters 346
selecting a specified word from
363
translating 376
trimming 377
updating 381
updating, based on byte unit 383
verifying 386
character sets
definition 9
specifying, for META declaration in
output 598
character strings 566
character variables
transcoding enabled for specified
variable 427
character-set encoding
for SAS session 577
characters
locating 562
CHARSET= option 598
China
monetary format 158, 195
code page 9
collating sequence 14
binary collation 14
language-specific 587
linguistic collation 15
overview 14
results of different sequences 14
compatibility
SAS string functions 279
compatible encodings 21
concatenation
character expressions 369
CPTDW informat 442
CPTWD informat. 443
Croatia
monetary format 166, 203
currency 6
yen 271
CVPBYTES option 605
Czech Republic
monetary format 159, 196

D
data set options
for transcoding 20
data sets
encoding 16
encoding support, by release 17
mixed encodings 43
suppressing transcoding 43
transcode attributes of variables 426
with a particular encoding 43
date format descriptors 388
date values
as a date 106
converting to specified locale 388
date and day of week 114
day of week 116
Japanese 102, 449
name and day of month 109
name of month 112
Taiwanese 101, 447
year 125
year and name of month 117
year and quarter 121
year and week 126
dates 5
date values as 106
DATESTYLE= system option
default values 636
datetime values
as a datetime 127
B8601DT format, ISO 8601 basic
notation, no time zone 82, 84
B8601TX format, ISO 8601 basic
notation, time zone offset 85
converting to specified locale 391, 394
day of week 140
day of week and datetime 139
E8601DT format, ISO 8601 basic
notation, no time zone 91, 93
E8601TX format, ISO 8601 basic
notation, time zone offset 94
name and day of month 131
name of month 135
name of month, day of month and year
129
time of day 137
with a.m. or p.m. 128
year 149
year and name of month 142
year and name of week 150
year and quarter 145
datetime-format descriptors 391
DBCS encoding
character data truncation 23
encoding values for transcoding data
643
removing leading blanks 350
trimming blanks and SO/SI 377
trimming trailing blanks 362
Denmark
monetary format 160, 197
DFLANG= system option 575
default values 636
double-byte characters
number in a character expression 328
code point discrepancies among
encodings 13
converting character data to 90
transferring data between ASCII and 22
Egypt
monetary format 162, 199
ENCODCOMPAT function 320
encoding 9
behavior in SAS sessions 17
compatibility for transcoding 21
converting one type of data to another
335
data set support by release 17
data sets 16
definition 10
for East Asian languages 11
input processing 17
mixed 43
output processing 17
overriding 41
reading and writing external files 17
standards organizations for 13
versus transcoding 10
encoding methods 9, 10
ENCODING option 612
ENCODING system option 577
encoding values 643
DBCS 643
for transcoding data 643
SBCS 643
Unicode 643
ENCODING= data set option 41
ENCODING= system option
default settings 636
Posix values 636
ENCODISVALID Function 321
Estonia
monetary format 161, 198
EURO format 96
EURO informat 444
euros
formats for 96, 99
EUROX format 99
EUROX informat 445
external files
BOM prefix on Unicode files 574
encoding and 17

E
E8601DXw. format 91
E8601LXw. format 93
E8601TXw. format 94
East Asian languages
encodings for 11
EBCDIC 11

F
Faroe Island
monetary format 160, 197
filerefs
limiting to eight characters 567
Finland
monetary format 163, 200
formats
  associating with variables 619
  international date and datetime formats 47
  language for international dates 575
NLS 47
France
  monetary format 163, 200
  functions
    by category 304
    K functions 279
    SAS string functions 279

G
Germany
  monetary format 163, 200
GETLOCENV function 322
GETPXLANGUAGE function 323
GETPXLCALOE function 324
GETPXREGION function 325
Greece
  monetary format 163, 200
Greenland
  monetary format 160, 197

H
Hebrew characters 88, 89
Hong Kong
  monetary format 165, 202
Hungary
  monetary format 167, 204

I
IBw.d informat 529, 530, 531, 533
illegal data 586
incompatible encodings 21
India
  monetary format 170, 207
Indonesia
  monetary format 168, 205
INENCODING= and OUTENCODING= option 615
informs
  associating with variables 619
  language for international dates 575
input processing 17
integer binary values, reading 529, 530, 531, 533
international date and datetime formats 47
international date formats and informs
  specifying language for 575
International Organization for
  Standardization (ISO) 13
Ireland
  monetary format 163, 200
ISO (International Organization for
  Standardization) 13
ISO 8601 date and time formats
  B8601DX format, basic datetime
    notation, time zone offset 82
  B8601LX format, basic datetime
    notation, time zone offset 84
  B8601TX format, basic time notation,
    time zone offset 85
  E8601DX format, basic datetime
    notation, time zone offset 91
  E8601LX format, basic datetime
    notation, time zone offset 93
  E8601TX format, basic time notation,
    time zone offset 94
ISO encodings 11
  8859 family 11
  Windows family 11
Israel
  monetary format 169, 206
Italy
  monetary format 163, 200
  item stores
    naming 589

J
Japan
  monetary format 171, 208
Japanese dates 102, 449

K
K functions 279
KCOMPARE function 326
KCOMPRESS function 327
KCOUNT function 328
KCOUNTC function 329
KCOUNTW function 331
KCOUNTX function 334
KCVT function 335
KFIND function 337
KFINDC function 339
KFINDW function 342
KINDEX function 346
KINDEXB function 347
KINDEXC function 348
KINDEXCB function 349
KLEFT function 350
KLENGTH function 353
KLOWCASE function 354
KPROPCASE function 355
KPROPCHAR function 358
KPROPDATA function 359
KREVERSE function 361
KRIGHT function 362
KSCAN function 363
KSCANX function 365
KSTRCAT function 369
KSTRIP function 370
KSUBSTR function 372
KSTRNB function 375
KTRANSLATE function 376
KTRIM function 377
KTURNCATE function 379
KUPCASE function 380
KUPCASE macro function 384
KUPDATE function 381
KUPDATEB function 383
KUPDATES function 384
KVERIFY function 386
KVERIFYB function 387

L
labels, associating with variables 619
language 5
language codes
   current two-letter code 323
language switching 7
   changing text language of ODS output 585
languages
   for international date informats and formats 575
Latvia
   monetary format 174, 211
left-aligning character expressions 350
length
   associating with variables 619
   of arguments 353
Liechtenstein
   monetary format 157, 194
line-feed characters 22
linguistic collation 15
linguistic sort keys 401
Lithuania
   monetary format 173, 210
locale 5
   best numerical notation based on 104
   converting date values to specified locale 388
   converting datetime values to specified locale 391
   converting time or datetime values to specified locale 394
language switching 7
   of SAS sessions 578
POSIX value for 324
specifying 6
   specifying at SAS invocation 6
   specifying during SAS session 7
LOCALE system option 578
LOCALE= system option values for 629
LOCALEDATA system option 579
localization 4
LOGLANGCHG system option 580
LOGLANGENG system option 581
long macro variables
   storing values in segments 567
lowercase characters
   changing uppercase characters to 35
   converting arguments to 354
LSWLANG system option 582
Luxembourg
   monetary format 163, 200

M
Macau
   monetary format 175, 212
macro variables
   storing long values in segments 567
Malaysia
   monetary format 177, 214
Malta
   monetary format 163, 200
MAPEBCDIC2ASCII system option 583
META declaration 598
Mexico
   monetary format 176, 213
MINGUO format 101
MINGUO informat 447
monetary formats
   Australia 153, 190
   Austria 200
   Belgium 163, 200
   Brazil 155, 192
   Bulgaria 154, 191
   Canada 156, 193
   China 158, 195
   Croatia 166, 203
   Czech Republic 159, 196
   Denmark 160, 197
   Egypt 162, 199
   Estonia 161, 198
   Faroe Island 160, 197
   Finland 163, 200
   France 163, 200
   Germany 163, 200
   Greece 163, 200
Greenland 160, 197
Hong Kong 165, 202
Hungary 167, 204
India 170, 207
Indonesia 168, 205
Ireland 163, 200
Israel 169, 206
Italy 163, 200
Japan 171, 208
Latvia 174, 211
Liechtenstein 157, 194
Lithuania 173, 210
Luxembourg 163, 200
Macau 175, 212
Malaysia 177, 214
Malta 163, 200
Mexico 176, 213
Netherlands 163, 200
New Zealand 179, 216
Norway 178, 215
Poland 180, 217
Portugal 163, 200
Puerto Rico 187, 224
Russia 181, 218
Singapore 183, 220
Slovenia 163, 200
South Africa 188, 225
South Korea 172, 209
Spain 163, 200
Sweden 182, 219
Switzerland 157, 194
Taiwan 186, 223
Thailand 184, 221
Turkey 185, 222
United Arab Emirates 152, 189
United Kingdom 164, 201
United States 187, 224

N
National Language Support
  See NLS (National Language Support)
National Language Support (NLS)
  formats 47
NENGO format 102
NENGO informat 449
Netherlands
  monetary format 163, 200
New Zealand
  monetary format 179, 216
NLBEST format 104
NDATE format 106
NDATE function 388
NDATE informat 450
NDATEL format 107
NDATEM format 108
NDATEMD format 109
NDATEMDL Format 110
NDATEMDM format 111, 133
NDATEMDS format 111
NDATEMN format 112
NDATEMYL format 118
NDATES format 113
NDATEW format 114
NDATEW informat 451
NDATEWN format 116
NDATEY format 117
NDATEYMM format 119
NDATEYMS format 120
NDATEYQ format 121
NDATEYQL format 122
NDATEYQM format 123
NDATEYQS format 124
NDATEYR format 125
NDATEYW format 126
LDATM format 127
LDATM informat 452
LDATMAP format 128
LDATMAP informat 453
LDATMDT format 129
LDATML format 130
LDATMM format 131
LDATMMD format 131
LDATMMDL format 132
LDATMMDMS format 134
LDATMNN format 135
LDATMS format 136
LDATMTM format 137
LDATMTZ format
datetime values 138
LDATMW format 139
LDATMW informat 454
LDATMWN format 140
LDATMWZ format
datetime values 141
LDATMYM format 142
LDATMYML format 143
LDATMYMM format 144
LDATMYMS format 144
LDATMYQ format 145
LDATMYQL format 146
LDATMYQM format 147
LDATMYQS format 148
LDATMYR format 149
LDATMYW format 150
LDATMZ format
datetime values 151
NLDECSEPARATOR system option 584
NLMNIAED format 152
NLMNIAED informat 455
NLMNIAUD format 153
NLMNIAUD informat 456
NLMNLLTL format 210
NLMNLLTLw.d informat 513
NLMNLLVL format 211
NLMNLLVLw.d informat 514
NLMNLMOP format 212
NLMNLMOPw.d informat 515
NLMNLMXN format 213
NLMNLMXNw.d informat 516
NLMNMYR format 214
NLMNMYRw.d informat 517
NLMNLNOK format 215
NLMNLNOKw.d informat 518
NLMNLNZD format 216
NLMNLNZDw.d informat 519
NLMNLPLN format 217
NLMNLPLNw.d informat 520
NLMNLRUB format 218
NLMNLRUBw.d informat 521
NLMNLSEK format 219
NLMNLSEKw.d informat 522
NLMNLUSD format 220
NLMNLUSDw.d informat 523
NLMNLTHB format 221
NLMNLTHBw.d informat 524
NLMNLTRY format 222
NLMNLTRYw.d informat 525
NLMNLUSD format 223
NLMNLUSDw.d informat 526
NLMNLUSD format 224
NLMNLUSDw.d informat 527
NLMNLZAR format 225
NLMNLZARw.d informat 528
NLMNY format 226
NLMNYI format 227
NLMNUM format 228
NLMNUMI format 230
NLNCT format 231
NLNCTI format 232
NLNCTIw.d informat 535
NLNCTN format 234
NLNCTP format 235
NLNCTw.d informat 534
NLNPVALUE format 236
NLS (National Language Support) 3
encoding 9
locale 5
transcoding 19
NLSTRMON format 237
NLSTRQTR format 238
NLSTRWK format 239
NLTIME format 241
NLTIMEw. informat 537
NODATM function 391
Norway
monetary format 178, 215
numbers 6
numeric data
Japanese dates 102
Taiwanese date values 101
yen 271
numeric values
truncating 379
numerical notation
best, based on locale 104

O
ODS output
changing language of the text 585
ODSCHARSET option 617
ODSLANGCHG system option 585
OPT= option, TRANTAB statement 622
output processing 17
OUTREP data set option 44
overriding encoding 41

P
paper size and measurement 6
PAPERSIZE= system option default values 636
Poland
monetary format 180, 217
Portugal
monetary format 163, 200
Puerto Rico
monetary format 187, 224

R
region codes
current two-letter code 325
remote applications
illegal data 586
right-alignment
character expressions 362
RSASIOTRANSERROR system option 586
Russia
monetary format 181, 218

S
SAS data sets
naming 589
SAS data views
naming 589
SAS language elements
using encoding values 643
SAS sessions
default character-set encoding 577
encoding behavior in 17
locale of 578
specifying locale during 7
SAS string functions
internationalization compatibility for 279
SASMSG function 395
SASMSGL function 398
SBCS encoding
encoding values for transcoding data 643
searching
for a word, by position in a string 564
for specific characters in character expression 348
for string of characters in character expression 346
segments
storing long macro variable values in 567
SETLOCALE function 404
shift out/shift in (SO/SI)
trimming from character expressions 377
Singapore
monetary format 183, 220
single-byte character set (SBCS)
encoding values for transcoding data 643
Slovenia
monetary format 163, 200
SO/SI (shift out/shift in)
trimming from character expressions 377
sort keys
linguistic 401
SORT option 599
SORT procedure
language-specific collating sequence for 587
SORTKEY function 401
SORTSEQ= system option 587
South Africa
monetary format 188, 225
South Korea
monetary format 172, 209
Spain
monetary format 163, 200
SQL procedure
language-specific collating sequence for 587
standards organizations 13
string functions
internationalization compatibility for 279
strings
length of 563
locating first character in 562
reducing length of 564
searching for words by position in 564
substring of a character string 566
substrings
extracting from an argument 372
extracting from an argument, based on byte position 375
of a character string 566
Sweden
monetary format 182, 219
Switzerland
monetary format 157, 194
system options
for transcoding 20
Taiwan
monetary format 186, 223
Taiwanese dates 101, 447
Thailand
monetary format 184, 221
time 6
time values
converting to specified locale 394
Time Zone 25
TIMESZONE= system option 588
tailing blanks
trimming 36
transcode attributes
of data set variables 426
TRANSCODE= Column Modifier on PROC SQL 618
TRANSCODE= option
ATTRIB statement 620
transcoding 10, 19
by specified translation table 420
compatible and incompatible encodings 21
considerations for 21
enabled for specified argument 428
enabled for specified character variable 427
encoding values for 643
line-feed characters 22
preventing 22
reasons for 20
SAS options for 20
suppressing 43
transferring data between EBCDIC and ASCII 22

T
Taiwan
monetary format 186, 223
Taiwanese dates 101, 447
Thailand
monetary format 184, 221
time 6
time values
converting to specified locale 394
Time Zone 25
TIMESZONE= system option 588
tailing blanks
trimming 36
transcode attributes
of data set variables 426
TRANSCODE= Column Modifier on PROC SQL 618
TRANSCODE= option
ATTRIB statement 620
transcoding 10, 19
by specified translation table 420
compatible and incompatible encodings 21
considerations for 21
enabled for specified argument 428
enabled for specified character variable 427
encoding values for 643
line-feed characters 22
preventing 22
reasons for 20
SAS options for 20
suppressing 43
transferring data between EBCDIC and ASCII 22

Index 711
versus encoding 10
transcoding errors 586
translating character expressions 376
translation tables
  applying to transport files 624
  transcoding by specified table 420
transport files
  applying translation tables to 624
TRANTAB function 420
TRANTAB statement
  UPLOAD procedure 622
TRANTAB-ODS option 618, 621
trimming trailing blanks 36
truncating numeric values 379
Turkey
  monetary format 185, 222
TYPE= option, TRANTAB statement 622
TZ=
  TZONEID Function 411
  TZONEDESTOFF function 417
  TZONEID function 411
  TZONENAME function 412
  TZONEOFF function 414
  TZONES2U function 415
  TZONESTTNAME function 418
  TZONESTTOFF function 419
  TZONEU2S function 419

U
Unicode 10
  BOM prefix on external files 574
  encoding values for transcoding data 643
  length of character unit 424
  length of display unit 425
Unicode Consortium 13
UNICODE function 421
UNICODEC function 423
UNICODELEN function 424
UNICODEWIDTH function 425
United Arab Emirates
  monetary format 152, 189
United Kingdom
  monetary format 164, 201
United States
  monetary format 187, 224
UPLOAD procedure
  TRANTAB statement 622
uploading files
  translation tables 622
uppercase characters
  changing to lowercase 35
  converting arguments to 380
UTF-16 11
UTF-32 11
UTF-8 10

V
VALIDMEMNAME system option 589
VALIDVARNAME= system option 592
variable names
  rules for valid names 592
variables
  associating formats with 619
  associating informats with 619
  labels 619
  length, associating with 619
  transcoding attributes of 426
  transcoding enabled for specified character variable 427
variant characters 14
VTRANSCODE function 426
VTRANSCODEX function 428

W
WEEKU format 266
WEEKV format 267
WEEKW format 269
Windows
  ISO encodings 11
words
  searching for, by position in a string 564

X
XMLENCODING option 622

Y
YEN format 271
YYWEEKU format 272
YYWEEKV format 273
YYWEEKW format 275
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.