<table>
<thead>
<tr>
<th>PART 3</th>
<th>Autocall Macros for NLS</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 7 • Dictionary of Autocall Macros for NLS</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>Autocall Macro Entries by Category</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>Dictionary</td>
<td>43</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 4</th>
<th>Data Set Options for NLS</th>
<th>47</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 8 • Dictionary of Data Set Options for NLS</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>Data Set Options by Category</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>Dictionary</td>
<td>49</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 5</th>
<th>Formats for NLS</th>
<th>53</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 9 • Overview to NLS Formats</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>International Date and Datetime Formats</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>Currency Representation</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>Exceptions for Date and Time Default Widths</td>
<td>67</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 6</th>
<th>Functions for NLS</th>
<th>279</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 11 • Internationalization Compatibility for SAS String Functions</td>
<td>281</td>
<td></td>
</tr>
<tr>
<td>Internationalization Compatibility for SAS String Functions</td>
<td>281</td>
<td></td>
</tr>
</tbody>
</table>

| Chapter 12 • VARCHAR Support in Functions | 301 |
| VARCHAR Data Type in String Functions | 301 |

<table>
<thead>
<tr>
<th>PART 7</th>
<th>Informats for NLS</th>
<th>403</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 13 • NLS Functions for SAS Only</td>
<td>305</td>
<td></td>
</tr>
<tr>
<td>Functions by Category</td>
<td>306</td>
<td></td>
</tr>
<tr>
<td>Dictionary</td>
<td>308</td>
<td></td>
</tr>
</tbody>
</table>

<p>| Chapter 14 • NLS Functions for SAS and the CAS Server | 375 |
| SAS Functions for CAS and SAS by Category | 375 |
| Dictionary | 376 |</p>
<table>
<thead>
<tr>
<th>Chapter 15 • Dictionary of Informats for NLS</th>
<th>405</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informats by Category</td>
<td>407</td>
</tr>
<tr>
<td>Dictionary</td>
<td>414</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 8 Macro Functions for NLS 531</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 16 • Dictionary of Macro Functions for NLS</td>
</tr>
<tr>
<td>Macro Functions by Category</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 9 System Options for NLS 543</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 17 • Dictionary of System Options for NLS</td>
</tr>
<tr>
<td>System Option Entries by Category</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 10 Options for Commands, Statements, and Procedures for NLS 567</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 18 • Dictionary of Command, Statement, and Procedure Option for NLS 569</td>
</tr>
<tr>
<td>Commands, Statements, and Procedures for NLS by Category</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 11 Values for Locale and Transcoding 597</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 19 • Values for the LOCALE= System Option</td>
</tr>
<tr>
<td>LOCALE= Values for PAPERSIZE and DFLANG Options</td>
</tr>
<tr>
<td>Chapter 20 • Encoding Values in SAS Language Elements</td>
</tr>
<tr>
<td>Overview to SAS Language Elements That Use Encoding Values</td>
</tr>
<tr>
<td>SBCS, DBCS, and Unicode Encoding Values for Transcoding Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 12 Appendixes 623</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix 1 • Time Zone IDs and Time Zone Names 625</td>
</tr>
<tr>
<td>Area: Africa</td>
</tr>
<tr>
<td>Area: America–North, Central, and South</td>
</tr>
<tr>
<td>Area: Antarctica</td>
</tr>
<tr>
<td>Area: Asia</td>
</tr>
<tr>
<td>Area: Atlantic</td>
</tr>
<tr>
<td>Area: Australia</td>
</tr>
<tr>
<td>Area: Miscellaneous</td>
</tr>
<tr>
<td>Area: Europe</td>
</tr>
<tr>
<td>Area: Pacific</td>
</tr>
</tbody>
</table>
Contents

Recommended Reading .................................................. 657
Index ................................................................. 659
Early Adopter Software

THIS DOCUMENTATION FOR AN EARLY ADOPTER PRODUCT IS A PRELIMINARY DRAFT AND IS PROVIDED BY SAS INSTITUTE INC. ("SAS") ON AN "AS IS" BASIS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. SAS does not warrant that this documentation is complete, accurate, or similar to that which may be released to the general public, or that any such documentation will be released. The company shall not be liable whatsoever for any damages arising out of the use of this documentation, including any direct, indirect, or consequential damages. SAS reserves the right to alter or abandon use of this documentation at any time.
Early Adopter Software
Part 1

NLS Concepts

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

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

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

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

Chapter 5
Time Zones ................................................................. 25
Chapter 1
National Language Support (NLS)

Overview to National Language Support

National Language Support (NLS) is a set of features that enable a software product to function properly in every global market for which the product is targeted. The SAS System 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 use the English language and use data that is formatted using the conventions that are observed in the United States. However, without NLS, these products might not work well for users in other regions of the world. NLS in SAS enables users in regions such as Asia and Europe to process data successfully in their native languages and environments.

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

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

Software applications that incorporate NLS can avoid dependencies on language-specific or cultural-specific conventions for software features such as:

- string manipulation
- character classifications
- character comparison rules
- code sets
- date and time formatting
- User interface
- message-text language
- 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 281.
Overview of Locale Concepts for NLS

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

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

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

Germany
03.10.16

Italy
3/10/16

United States
10/03/16

Language is part of a locale, but is not unique to any one locale. For example, Portuguese is spoken in Brazil as well as in Portugal, but the cultures are different. In Brazil and in Portugal, there are similarities in the formatting of data. Numbers are formatted using a comma (,) to separate integers from fractional values and a dot (.) to separate groups of digits to the left of the radix character. However, there are important differences, such as the currency symbols that are used in the two different locales. Portugal uses the Euro and requires the Euro symbol € while Brazil uses the Real that is represented by the two-character currency symbol R$.
Also, a country might have more than one official language. Canada has two official languages: English and French; two values can be specified for the LOCALE= system option: English_Canada and French_Canada.

Numbers, including currency, can have different representations. For example, the decimal separator, or radix character, is a dot (.) in some regions and a comma (, ) in other regions. The thousands separator can be a dot, comma, or even a space in some regions. Monetary conventions likewise vary between locales; for example, a dollar sign or a yen sign might be attached to a monetary value.

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

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

### Specifying a Locale

**How Locale Is Specified at SAS Invocation**

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

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

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

Default values for the LOCALE= option are the same under each operating environment. For details, see “LOCALE= Values for PAPERSIZE and DFLANG Options” on page 599.

The English_UnitedStates value for LOCALE= causes the following options to be implicitly set to the specified default values 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 via the LOCALE= option. For details, see “PAGESIZE= System Option” in SAS Viya System Options: Reference.

**How Locale Is Specified during a SAS Session**

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

- DFLANG=
- PAPERSIZE=

*Note:* The PAPERSIZE option is affected only if it is set to LOCALE. For more information see “PAGESIZE= System Option” in SAS 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 causes the following options to be implicitly reset during the SAS session to reflect the changed value of the LOCALE= system option:

- DFLANG=Italian
- PAPERSIZE=A4

**Language Switching**

SAS messages are displayed in the language that is specified by the settings in the SAS configuration file during start-up. You can view SAS messages in another language by using the Language Switching feature. You can access the Language Switching feature with the ODSLANGCHG system option. If ODSLANGCHG is enabled, then the value of the LOCALE system option determines the language for procedure output, user interface elements, and ODS fonts. If ODSLANGCHG is disabled, then messages appear in the language that is set during start-up. This feature is supported in 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 557. The LSWLANG system option specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation. LSWLANG 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 552.

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 554.
The LOGLANGENG option is a toggle option that overrides LOGLANGCHG and LSWLANG and sets them to \texttt{LOGLANGCHG=ON} and \texttt{LSWLNG=EN}. As a result, the LOG output is in English, and the NL format output has no change. This option changes the setting of ODSLANGCHG. If \texttt{ODSLANGCHG=OFF}, the system message language for ODS output is determined by SAS configuration. If \texttt{ODSLANGCHG=ON}, all messages are in English because of the LSWLANG setting. For more information, see “LOGLANGENG System Option” on page 553.

For more information about Language Switching, see \textit{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 (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. In order for SAS Viya that 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**

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>Latin 1</td>
<td>US and Western European</td>
</tr>
<tr>
<td>ISO 8859-2</td>
<td>Latin 2</td>
<td>Central and Eastern European</td>
</tr>
<tr>
<td>ISO 8859-3</td>
<td>Latin 3</td>
<td>Southern European, Maltese, and Esperanto</td>
</tr>
</tbody>
</table>
## ISO Standard

<table>
<thead>
<tr>
<th>ISO Standard</th>
<th>Name of Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 multibyte encoding</td>
<td>Japan</td>
</tr>
</tbody>
</table>

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

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

International Organization for Standardization (ISO) promotes the development of standardization and related activities 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 that develops and promotes the Unicode standard, which provides a unique number for every character.

Code Point Discrepancies among EBCDIC Encodings

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

Table 3.3 Variant Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>1047</th>
<th>838</th>
<th>870</th>
<th>1025</th>
<th>1141</th>
<th>1142</th>
<th>1143</th>
<th>1144</th>
<th>1145</th>
<th>1146</th>
<th>1147</th>
<th>1148</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>5A</td>
<td>5A</td>
<td>4F</td>
<td>4F</td>
<td>4F</td>
<td>4F</td>
<td>4F</td>
<td>BF</td>
<td>5A</td>
<td>4F</td>
<td>4F</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>7B</td>
<td>7B</td>
<td>7B</td>
<td>7B</td>
<td>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>E0</td>
<td>E0</td>
<td>E0</td>
<td>E0</td>
<td>EC</td>
<td>E0</td>
<td>71</td>
<td>48</td>
<td>E0</td>
<td>E0</td>
<td>48</td>
<td>E0</td>
</tr>
<tr>
<td>[</td>
<td>AD</td>
<td>49</td>
<td>4A</td>
<td>4A</td>
<td>63</td>
<td>9E</td>
<td>B5</td>
<td>90</td>
<td>4A</td>
<td>B1</td>
<td>90</td>
<td>4A</td>
</tr>
<tr>
<td>]</td>
<td>BD</td>
<td>59</td>
<td>5A</td>
<td>5A</td>
<td>FC</td>
<td>9F</td>
<td>9F</td>
<td>51</td>
<td>5A</td>
<td>BB</td>
<td>B5</td>
<td>5A</td>
</tr>
<tr>
<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>
<td>5F</td>
</tr>
<tr>
<td>`</td>
<td>79</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>
</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 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
• 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>
<tr>
<td>Binary</td>
<td>Translation Table</td>
<td>Encoding Value</td>
<td>Linguistic</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>Aztec</td>
<td>azimuth</td>
<td>Aztec</td>
<td>Aaron</td>
</tr>
<tr>
<td>Zeus</td>
<td>Aaron</td>
<td>Zeus</td>
<td>azimuth</td>
</tr>
<tr>
<td>aardvark</td>
<td>Aztec</td>
<td>aardvark</td>
<td>Aztec</td>
</tr>
<tr>
<td>azimuth</td>
<td>cote</td>
<td>azimuth</td>
<td>cote</td>
</tr>
<tr>
<td>cote</td>
<td>coté</td>
<td>cote</td>
<td>coté</td>
</tr>
<tr>
<td>coté</td>
<td>côte</td>
<td>coté</td>
<td>coté</td>
</tr>
<tr>
<td>côte</td>
<td>côté</td>
<td>côte</td>
<td>côté</td>
</tr>
<tr>
<td>côté</td>
<td>zebra</td>
<td>côté</td>
<td>zebra</td>
</tr>
<tr>
<td>zebra</td>
<td>zèbre</td>
<td>zèbre</td>
<td>zèbre</td>
</tr>
<tr>
<td>zèbre</td>
<td>Zeus</td>
<td>zèbre</td>
<td>Zeus</td>
</tr>
</tbody>
</table>

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

The second column shows the results of specifying a translation table that alternates the ordering of lowercase and uppercase characters. If you use the translation table, the word aardvark appears before Zeus. However, the word azimuth appears before Aaron because the translation table assigns a weight value to the lowercase character a that is less than the weight value of the uppercase character A. In addition, accents are sorted from left to right. For example, coté comes before 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, the following statements cause the SORT procedure to collate 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, local conventions such as data formatting, and culture for a geographical region. You can modify the algorithm by specifying options in parentheses following the LINGUISTIC keyword. For example, you can specify a different locale; you can specify the CASE_FIRST= option to collate lowercase characters before uppercase characters, or vice versa; and so on. Generally, it is not necessary to specify options, because the ICU associates defaults with the various languages and locales. For more information about the linguistic options, see the SORTSEQ= option in “Collating Sequence Option” on page 571 or the SORTSEQ= option in the PROC SORT statement in *SAS Viya Utility Procedures Guide*.

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

---

**Determining the Encoding of a SAS Data Set**

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

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

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

Here is the output from the PROC CONTENTS code. The encoding is us-ascii.

<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_IAX64</td>
<td></td>
</tr>
</tbody>
</table>
| Encoding      | us-ascii ASCII (ANSI) | }
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, 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, which specifies a data representation that is different from the current session, 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 52.

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

Input Processing

For input (read) processing in SAS, 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 in it, 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:

    filename in 'external-file'
    encodings='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, then moved to SAS Viya that uses UTF-8 encoding. When the file is processed on SAS Viya, the data is remapped from the Latin1 encoding to UTF-8 encoding.

Transcoding does not translate between languages; transcoding remaps characters.

In order to dynamically transcode data between operating environments that use different encodings, an explicit encoding value must be specified. For details, see Chapter 20, “Encoding Values in SAS Language Elements,” on page 613.
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>in a DATA step</td>
</tr>
<tr>
<td>INENCODING=</td>
<td>LIBNAME statement</td>
</tr>
<tr>
<td>ODSCHARSET=</td>
<td>LIBNAME statement for XML</td>
</tr>
<tr>
<td>ODSTRANTAB=</td>
<td>LIBNAME statement for XML</td>
</tr>
<tr>
<td>OUTENCODING=</td>
<td>LIBNAME statement</td>
</tr>
<tr>
<td>XMLENCODING=</td>
<td>LIBNAME statement for XML</td>
</tr>
</tbody>
</table>

For a list of supported encoding values to use for these options, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 613.
Transcoding Considerations

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

- Encodings 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 DBCS to an SBCS. Therefore, transcoding can result in character value truncation.
- If an error occurs during transcoding such 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

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

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

**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 but rather you might want to create a SAS library that contains data in mixed encodings. 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= options 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.
Avoiding Character Data Truncation By Using the CVP Engine

When you specify the ENCODING= data set option, the encoding for the output data set might require more space than the original data set. For example, when writing DBCS data in a Windows environment using the UTF-8 encoding, each DBCS character might require three bytes. To avoid data truncation, each variable must have a width that is 1.5 times greater than the width of the original data.

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

Character data truncation can occur when the number of bytes for a character in one encoding is different from the number of bytes for the same character in another encoding, such as when a single-byte character set (SBCS) is transcoded to a double-byte character set (DBCS) or to a multi-byte character set (MBCS). An SBCS represents each character in one byte, and a DBCS represents each character in two bytes. An MBCS represents characters in a varying length from one to four bytes. For example, when transcoding from WLatin2 to a Unicode encoding, such as UTF-8, the variable lengths (in bytes) might not be sufficient to hold the values, and the result is character data truncation.

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

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

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

For example, the following LIBNAME statement explicitly assigns the CVP engine. Character variable lengths are increased using the default expansion, which 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. Notes are 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.

For more information and examples, see “CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options” on page 577.
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
Specify the local time when a time zone is not specified by the TIMEZONE= system option.

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

Time zone ID
Specifies a region and area separated by a forward slash (/). Asia/Tokyo is a time zone ID. Time zone IDs are compatible with Java time zones. For a list of time zone IDs, see Appendix 1, “Time Zone IDs and Time Zone Names,” on page 625.

Time zone name
Specifies three or four characters that indicate a time zone. For a list of time zone names, see Appendix 1, “Time Zone IDs and Time Zone Names,” on page 625.

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

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

Universal Coordinated Time (UTC)
Specify the time at the zero meridian, near Greenwich, England. UTC is a datetime value that uses the ISO 8601 basic form yyyymmddThh:mm:ss+|-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 1, “Time Zone IDs and Time Zone Names,” on page 625.

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:

\[
t\text{stamp='2013-05-17T09:15:30–05:00'dt;}
\]

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 \( \text{TIME( )} \) function, the \( \text{TODAY( )} \) function, the \( \text{DATE( )} \) function, and the \( \text{DATETIME( )} \) 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 \( \text{TIMEZONE=} \) system option:

\[
\text{options timezone='asia/tokyo';}
\]

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 1, “Time Zone IDs and Time Zone Names,” on page 625. For more information, see “\( \text{TIMEZONE=} \) System Option” on page 560.

**Determine a Time Zone ID Offset**
You use the \( \text{TZONEOFF( )} \) function to determine a time zone name or time zone ID offset:

- The \( \text{TZONEOFF( )} \) function returns the time zone offset for the current time zone.
- The \( \text{TZONEOFF('time-zone-id')} \) 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:

\[
\text{data _null_;}
\text{ o1=tzoneoff();}
\text{ o2=tzoneoff('asia/tokyo');}
\text{ put o1 time.;}
\text{ put o2 time.;}
\text{ run;}
\]

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 \( \text{ABS( )} \) function:

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

For more information, see “\( \text{TZONEOFF Function} \)” on page 360.

**Determine a Time Zone ID or Time Zone Name**
Use these functions to determine a time zone name or time zone ID:

- The \( \text{TZONEID( )} \) function returns the current time zone ID.
• The TZNENAME( ) function returns the current time zone name based on a time
zone ID and a SAS datetime value or based on a SAS datetime value only.

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

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

Here is the output:

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

For more information, see “TZONEID Function” on page 357 and “TZONENAME
Function” on page 359.

**Convert Datetime Values between SAS and UTC**

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

• The TZNES2U( ) converts a SAS datetime value to a UTC datetime value.

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

Here are some examples of using these functions:

```sas
options timezone='est';
data _null_;  
put ' The time zone is EST';
diff=abs(tzoneoff('america/new_york') - tzoneoff('europe/london'));
put ' New York-London difference: ' diff time.;
diff=abs(tzoneoff('america/new_york') - tzoneoff('asia/tokyo'));
put ' New York-Tokyo difference: ' diff time.;
put ' The SAS datetime is 2013-03-15T09:15:00+00:00 ';  
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';
```
### Write ISO 8601 Time and Datetime Values Based on the Time Zone

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

- The `B8601TX.` and `E8601TX.` formats convert a UTC time value to a user local time and write the time value with a time zone offset.
- The `B8601DX.` and `E8601DX.` formats convert a UTC datetime value to a user local time and write the datetime value with a time zone offset.
- The `B8601LX.` and `E8601LX.` formats write a local datetime value using the UTC offset for the user local time zone.
This program formats time and datetime values based on the time zone ID Australia/Sydney:

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

Here is the output:

```
Time B8601TX:   043340+1000
Time E8601TX:04:33:40+10:00
UTC B8601DX:      20130318T123022+1100
UTC E8601DX: 2013-03-18T12:30:22+11:00
UTC B8601LX:      20130318T013022+1100
UTC E8601LX: 2013-03-18T01:30:22+11:00
```

For more information, see “B8601DX Format” on page 84, “B8601LX Format” on page 86, “B8601TX Format” on page 87, “E8601DX Format” on page 93, “E8601LX Format” on page 94, and “E8601TX Format” on page 96.

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

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

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

Here is the output:

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

For more information, see “NLDATMTZ Format” on page 139, “NLDATMZ Format” on page 152, and “NLDATMWZ Format” on page 142.
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 */
derpart='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:

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

Migration to UTF-8

Chapter 6
Migrating Data to UTF-8
Chapter 6
Migrating Data to UTF-8

SAS Viya and UTF-8 Encoding ................................................. 35
Determining the Encoding of a Data Set .................................. 36
Using CEDA to Read Data ..................................................... 36
Using CVP to Compensate for Space Issues and Save as a UTF-8 Encoded SAS Data Set .......................................................... 37
Converting Indexes and Constraints to UTF-8 .......................... 38
Converting Format Catalogs to UTF-8 ..................................... 38
Reading External Files ......................................................... 39
Processing Multi-Byte Character Data ....................................... 39

SAS Viya and UTF-8 Encoding

UTF-8 is the only SAS session encoding supported by SAS Viya. UTF-8 is a multi-byte encoding and part of the Unicode coded character set. UTF-8 is becoming the preferred and most-used encoding, and it is the recommended encoding for using Unicode with Linux-style operating systems.

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

Because SAS Viya supports only the UTF-8 session encoding, you must be aware of the encoding of your legacy data sets. You might have SAS 9 data sets that are not in UTF-8 encoding. You need to determine the encoding of your data and take steps to convert the data to UTF-8.

1. Determine the encoding of your data set. See “Determining the Encoding of a Data Set” on page 36.

2. Determine if CEDA is the appropriate conversion method. See “Using CEDA to Read Data” on page 36.

3. Determine if CVP is an appropriate conversion method and if your data has space issues. See “Using CVP to Compensate for Space Issues and Save as a UTF-8 Encoded SAS Data Set” on page 37.

5. Convert Format Catalogs to UTF-8. See “Converting Format Catalogs to UTF-8” on page 38.

6. Read external files. See “Reading External Files” on page 39.

---

**Determining the Encoding of a Data Set**

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

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

Here is the output from the PROC CONTENTS code. The encoding is us-ascii.

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>SASHELP.CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
</tr>
<tr>
<td>Created</td>
<td>01/17/2016 20:12:45</td>
</tr>
<tr>
<td>Last Modified</td>
<td>01/17/2016 20:12:45</td>
</tr>
<tr>
<td>Protection</td>
<td></td>
</tr>
<tr>
<td>Data Set Type</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Student Data</td>
</tr>
<tr>
<td>Data Representation</td>
<td>SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64</td>
</tr>
<tr>
<td>Encoding</td>
<td>us-ascii ASCII (ANSI)</td>
</tr>
</tbody>
</table>

---

**Using CEDA to Read Data**

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

When the source and/or session encodings are not the same, CEDA is used to read and convert the data. When the UTF-8 session reads a data set created with a latin1 encoding, you will see the following note in your log:

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

If all of the characters in the data sets are ASCII characters, you should have no problem using the data because the first 128 characters of Unicode correspond one-to-one with ASCII. There is a performance cost because the data set encoding does not match the session encoding and CEDA must transcode every record.

You do not have to use CEDA if your data set only contains ASCII characters. You can change the encoding of your data set to ASCIIANY to prevent transcoding. You use the
CORRECTENCODING option on the PROC DATASETS MODIFY statement to change the encoding that is stored in the data set header. Here is an example of specifying ASCIIANY.

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

Note: If the data set contains non-ASCII characters, you must continue using CEDA or convert the data set to UTF-8. If you used a Microsoft product to create your data, any quote or dash characters in the data might have been converted to the smart quotes or dashes. Those are not ASCII characters. A transcoding error might be displayed similar to the one in “Using CVP to Compensate for Space Issues and Save as a UTF-8 Encoded SAS Data Set” on page 37. These characters require more than one byte when converted from Latin1 to UTF-8.

---

Using CVP to Compensate for Space Issues and Save as a UTF-8 Encoded SAS Data Set

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

```
ERROR: Some character data was lost during transcoding in the data set ZHOLD.CARS. Either the data contains characters that are not representable in the new encoding or truncation occurred during transcoding.
```

This error usually means that there is not enough space in one or more character columns in the data set's observation buffer to convert the data to UTF-8. To solve the problem, use the Character Variable Padding (CVP) libname engine to read the data set. The CVP engine adds space to the character columns. By default, the column length is multiplied by 1.5. Use the CVP LIBNAME statement option CVPMULT= to control the amount of padding. Here is an example of using CVP with the default multiplier.

```
/* Libname for the same data set using the CVP engine. */
/* The CVP engine widens the character columns. */
libname mylib CVP "path to data sets";
```

Libraries accessed with the CVP engine are READONLY. If you want to save a permanent copy of the data, you need to create a new data set. You will need a second LIBNAME statement pointing to the location of your library. The following example uses PROC COPY to save the data. Note that you must use the NOCLONE option on the PROC COPY statement. Otherwise, the attributes from the original data set are duplicated in the new copy.

```
libname zhold cvp "/path to data";
libname zhnew "mylib";

proc copy in=zhold out=zhnew noclone;
select class;
run;

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

For more information see “Avoiding Character Data Truncation By Using the CVP Engine” on page 23 and “LIBNAME Statement, CVP Engine” in SAS Viya Statements: Reference.
Converting Indexes and Constraints to UTF-8

Indexes and integrity constraints that are created with a different encoding cannot be used by CEDA. Here are the steps to recreate indexes and constraints:

1. Specify the OUT2= option on PROC CONTENTS on the SAS 9 system using the original encoding to capture the definitions and encoding of the indexes and constraints in the data sets.

2. The data sets created by this option contain a variable named RECREATE that contains the exact SAS code needed in a PROC DATASETS step with a MODIFY statement to recreate the index or constraint recorded in that row of the data set.

3. Copy the original data set and the data set created by PROC CONTENTS to SAS Viya, and use the values in the RECREATE variable in a PROC DATASETS statement to recreate the indexes and/or constraints.

Converting Format Catalogs to UTF-8

SAS catalogs are SAS files that store different types of information in smaller units called catalog entries. Some catalog entries contain system information such as key definitions. Other catalog entries contain application information such as window definitions, help windows, formats, informats, macros, or graphics output. SAS catalog entries are generally accessed automatically by SAS when the information stored in them is required for processing.

Catalog entries containing formats must be converted to UTF-8 encoding. Also, if a non-UTF-8 encoding data set has a user defined format associated with a column, you must do the following:

- remove the format
- convert the data set to UTF-8
- associate the format with the column in the data set of the UTF-8 version.

You must recreate the catalog on the new operating system. For example, compiled SAS Macro catalogs are recreated by running the source code for the macro compilation on the new platform. For Format catalogs, you must run PROC FORMAT with the CNTLOUT option, in the SAS 9 latin1 environment which creates the data set containing the format definition. The data set is then transcoded to UTF-8 in the SAS Viya environment using a CVP libname, unless it already has enough space to prevent truncation, and the UTF8 data set is used in a PROC FORMAT step with a CNTLIN option. The resulting new format entry is UTF8 and correct for non-ASCII characters.

Programs which create permanent or temporary formats that have affected characters must also be transcoded in order to correctly create the formats when they are run in the new environment.
Reading External Files

SAS reads and writes external files using the current session encoding, and assumes that the external file uses the same encoding as the SAS session. When a file contains character data and its encoding is different from the SAS session encoding, use the ENCODING= option on the FILENAME, INFILE, or FILE statement to specify the file encoding. SAS can transcode the data from its original encoding to the current SAS session encoding. Here is an example:

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

Processing Multi-Byte Character Data

The original SAS string functions, such as LENGTH and SUBSTR, are byte-based. That means the functions assume that one character is always equal to one byte and return results that are based on the bytes in the string. If the data in your UTF-8 SAS environment contains multi-byte characters, LENGTH, SUBSTR and other string functions might produce unexpected results. For example, LENGTH returns the number of bytes in the specified string, not the number of characters.

SAS provides a set of string functions, called K functions, that resolve the issues for multi-byte data. The K functions are character-based, which means they do not make assumptions about the size of the characters in the string. For example, in SAS Viya KLENGTH returns the number of UTF-8 characters in the string. The position and length specified to KSUBSTR are interpreted as the character position and the number of characters to return. For more information about K Functions and which string functions support multi-byte data, see “Internationalization Compatibility for SAS String Functions” on page 281.
Part 3

Autocall Macros for NLS

Chapter 7

Dictionary of Autocall Macros for NLS .............................. 43
Chapter 7
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. 43)</td>
<td>Change uppercase characters to lowercase.</td>
</tr>
<tr>
<td></td>
<td>%KTRIM and %QKTRIM Autocall Macros (p. 44)</td>
<td>Trim trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%KVERIFY Autocall Macro (p. 44)</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 4

Data Set Options for NLS

Chapter 8
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</td>
<td>Overrides the encoding to use for reading or writing a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>(p. 49)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OUTREP= Data Set Option</td>
<td>Specifies the data representation for the output SAS data set.</td>
</tr>
<tr>
<td></td>
<td>(p. 52)</td>
<td></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 582
- “INENCODING= and OUTENCODING= Options” on page 585

System Options:

- “ENCODING System Option” on page 549
- “LOCALE System Option” on page 550

---

**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 5

Formats for NLS

Chapter 9
  Overview to NLS Formats ........................................ 55

Chapter 10
  Dictionary of Formats for NLS ................................. 69
Chapter 9
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 9.1  International Date and Datetime Formats

<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>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>10</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>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>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>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>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>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>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>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>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>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>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>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>DATE</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td></td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td></td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td></td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td></td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>29</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></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>NLDATEmn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>DOWNAME.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>MONNAME.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>MONYY.</td>
<td>NLDATEmn.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>WEEKDATX.</td>
<td>NLDATewn.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>WEEKDAY.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>WORDDATX.</td>
<td>NLDATEmn.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>DATE.</td>
<td>NLDATEmn.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>DOWNAME.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>MONNAME.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>MONYY.</td>
<td>NLDATEmn.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>WEEKDATX.</td>
<td>NLDATewn.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>WEEKDAY.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>WORDDATX.</td>
<td>NLDATEmn.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>DATE.</td>
<td>NLDATEmn.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>DOWNAME.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>MONNAME.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>MONYY.</td>
<td>NLDATEmn.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>WEEKDATX.</td>
<td>NLDATewn.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>WEEKDAY.</td>
<td>NLDATewn.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>WORDDATX.</td>
<td>NLDATewn.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>DATE.</td>
<td>NLDATEmn.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>DOWNAME.</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></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td></td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Spanish (ESP)</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>Slovenian (SLO)</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>Swedish (SVE)</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>
</tbody>
</table>
Currency Representation

Overview to Currency

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

U.S. Dollars

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

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

Example:

$12,345.00

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

Example:

$12.345,00

Limitations of the DOLLAR formats and informats are:

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

Euros

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

Example:

options locale=English_UnitedKingdom;

data _null_;
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 details about the FORMAT procedure, see *SAS Viya Utility Procedures Guide*.

**Example Code 9.1  SAS Code That Customizes Currency Representations**

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

  picture sfr low-<0='0,000,000,009.00'
    (prefix='-SFr.' mult=100)
  0–high='0,000,00,009.00'
    (prefix='SFr.' mult=100);

  picture bpd low-<0='0,000,000,009.00'
    (prefix='-BPd.' mult=100)
  0–high='0,000,00,009.00'
    (prefix='BPd.' mult=100);
run;

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 requires a programming solution.
Localized National and International Currency Representations

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

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

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

$12,345.00

Selected national currency representations follow:

<table>
<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. Whereas French_France uses no thousands separator but uses a comma as a decimal separator, German_Germany and Spanish_Spain use a dot as a thousands separator and a comma as a decimal separator.

Localized International currency representation
conforms to ISO standard 4217. International forms are specified using the 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.

Example:
```
options locale=english_UnitedStates;
data _null_;
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. Whereas the EMU countries put the currency code after the currency, English_UnitedStates, German_Switzerland, and Spanish_Venezuela precede the currency with the ISO code.

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

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

Unique National and International Monetary Representations

The NLMNLISOw.d and NLMNIISOw.d formats and informats were introduced to uniquely represent each currency without having to also use the LOCALE= option to specify the locale. 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 NLMNLISOw.d formats and informats. In the following example, USD is the ISO currency code for American dollars.

Note: When using the NLMNLISOw.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.

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

Output: 
**€12,345.00**

**Table 9.4** Unique Currency Representations by ISO Currency Code

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

A currency symbol or a currency code precedes most currencies. Also 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 3-letter ISO currency code.

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

**Note:** When using the NLMNIISOw.d formats and informats, you 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 9.5  International Currency Representations by ISO Currency Code

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

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

Example: Representing Currency in National and International Formats

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

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

```sas
data curr;
  input ex_date mmddyy. usd aud hkd jpy sgd 12.2;
datalines;
  061704 1.00000 1.45349 7.79930 110.110 1.71900
;
proc print data=curr noobs label;
  var ex_date usd aud hkd jpy sgd;
  format ex_date mmddyy. usd nlmnlusd15.2 aud nlmnlaud15.2 hkd
  nlmnlhkd15.2 jpy nlmnljpy15.2 sgd nlmnlsgd15.2;
```
label ex_date='Date' usd='US' aud='Australia' hkd='Hong Kong' jpy='Japan' sgd='Singapore';

title 'Exchange Rates for Selected Asian-Pacific Countries (Localized Currency Codes)';

proc print data=curr noobs label;
  var ex_date usd aud hkd jpy sgd;
  format ex_date mmddyy. usd nlmniusd15.2 aud nlmniaud15.2 hkd nlmnihkd15.2 jpy nlmnijpy15.2 sgd nlmnisgd15.2;
run;

1. These exchange rates, which were effective June 17, 2004, are specified as data in the SAS program.
2. These NLMNLISO formats are applied to each of the numeric data items that are specified in the INPUT statement. These formats show currencies in the appropriate national formats.
3. These NLMNIISO formats are applied to each of the numeric data items that are specified in the INPUT statement. These formats show currencies in the appropriate international formats.

Figure 9.1 National and International Format Output

Exceptions for Date and Time Default Widths

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></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>
</tbody>
</table>
Chapter 10
Dictionary of Formats for NLS

Categories of NLS Formats ................................................................. 72
Dictionary ................................................................. 84
  B8601DX Format ................................................................. 84
  B8601LX Format ................................................................. 86
  B8601TX Format ................................................................. 87
  BESTDOTX Format ............................................................... 88
  $CPTDW Format ................................................................. 90
  $CPTWD Format ................................................................. 91
  $EBCDIC Format ................................................................. 92
  E8601DX Format ................................................................. 93
  E8601LX Format ................................................................. 94
  E8601TX Format ................................................................. 96
  EURO Format .................................................................... 97
  EUROX Format ................................................................. 100
  MINGUO Format ............................................................... 102
  NENGO Format ................................................................. 104
  NLBEST Format ............................................................... 105
  NLDATE Format ............................................................... 107
  NLDATEL Format ............................................................. 108
  NLDATEM Format ............................................................. 109
  NLDATEMD Format .......................................................... 110
  NLDATEMDL Format ......................................................... 111
  NLDATEMDM Format ......................................................... 111
  NLDATEMS Format .......................................................... 112
  NLDATEMN Format .......................................................... 113
  NLDATES Format ............................................................. 114
  NLDATENW Format .......................................................... 115
  NLDATENW Format .......................................................... 116
  NLDATEYM Format .......................................................... 118
  NLDATEYML Format ......................................................... 119
  NLDATEYMM Format ......................................................... 120
  NLDATEYMS Format ......................................................... 121
  NLDATEYQ Format ........................................................... 122
  NLDATEYQL Format ......................................................... 123
  NLDATEYQM Format ........................................................ 124
  NLDATEYQS Format ........................................................ 125
  NLDATEYR Format ........................................................... 126
  NLDATEYW Format ........................................................... 127
  NLDATM Format .............................................................. 128
  NLDATMAP Format .......................................................... 129
  NLDATMDT Format .......................................................... 130
<table>
<thead>
<tr>
<th>Format</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLMNITHB Format</td>
<td>185</td>
</tr>
<tr>
<td>NLNITRY Format</td>
<td>186</td>
</tr>
<tr>
<td>NLNITWD Format</td>
<td>187</td>
</tr>
<tr>
<td>NLNIUSD Format</td>
<td>188</td>
</tr>
<tr>
<td>NLNIZAR Format</td>
<td>189</td>
</tr>
<tr>
<td>NLMLAED Format</td>
<td>190</td>
</tr>
<tr>
<td>NLMLAUD Format</td>
<td>191</td>
</tr>
<tr>
<td>NLMLBGN Format</td>
<td>192</td>
</tr>
<tr>
<td>NLMLBRL Format</td>
<td>193</td>
</tr>
<tr>
<td>NLMLCAD Format</td>
<td>194</td>
</tr>
<tr>
<td>NLMLCHF Format</td>
<td>195</td>
</tr>
<tr>
<td>NLMLCNY Format</td>
<td>196</td>
</tr>
<tr>
<td>NLMLCZK Format</td>
<td>197</td>
</tr>
<tr>
<td>NLMLDKK Format</td>
<td>198</td>
</tr>
<tr>
<td>NLMLEEK Format</td>
<td>199</td>
</tr>
<tr>
<td>NLMLEGP Format</td>
<td>200</td>
</tr>
<tr>
<td>NLMLEUR Format</td>
<td>201</td>
</tr>
<tr>
<td>NLMLGBP Format</td>
<td>202</td>
</tr>
<tr>
<td>NLMLHKD Format</td>
<td>203</td>
</tr>
<tr>
<td>NLMLHRK Format</td>
<td>204</td>
</tr>
<tr>
<td>NLMLHUF Format</td>
<td>205</td>
</tr>
<tr>
<td>NLMLIDR Format</td>
<td>206</td>
</tr>
<tr>
<td>NLMLIILS Format</td>
<td>207</td>
</tr>
<tr>
<td>NLMLINR Format</td>
<td>208</td>
</tr>
<tr>
<td>NLMLJPY Format</td>
<td>209</td>
</tr>
<tr>
<td>NLMLKRW Format</td>
<td>210</td>
</tr>
<tr>
<td>NLMLLTL Format</td>
<td>211</td>
</tr>
<tr>
<td>NLMLLLVL Format</td>
<td>212</td>
</tr>
<tr>
<td>NLMLMOP Format</td>
<td>213</td>
</tr>
<tr>
<td>NLMLMXN Format</td>
<td>214</td>
</tr>
<tr>
<td>NLMLMYR Format</td>
<td>215</td>
</tr>
<tr>
<td>NLMLNOK Format</td>
<td>216</td>
</tr>
<tr>
<td>NLMLNZD Format</td>
<td>217</td>
</tr>
<tr>
<td>NLMLPLN Format</td>
<td>218</td>
</tr>
<tr>
<td>NLMLRUB Format</td>
<td>219</td>
</tr>
<tr>
<td>NLMLSEK Format</td>
<td>220</td>
</tr>
<tr>
<td>NLMLSGD Format</td>
<td>221</td>
</tr>
<tr>
<td>NLMLTHB Format</td>
<td>222</td>
</tr>
<tr>
<td>NLMLTRY Format</td>
<td>223</td>
</tr>
<tr>
<td>NLMLTWD Format</td>
<td>224</td>
</tr>
<tr>
<td>NLMLUSD Format</td>
<td>225</td>
</tr>
<tr>
<td>NLMLZAR Format</td>
<td>226</td>
</tr>
<tr>
<td>NLMY Format</td>
<td>227</td>
</tr>
<tr>
<td>NLNYI Format</td>
<td>228</td>
</tr>
<tr>
<td>NLNUM Format</td>
<td>229</td>
</tr>
<tr>
<td>NLNUMI Format</td>
<td>231</td>
</tr>
<tr>
<td>NLPCT Format</td>
<td>232</td>
</tr>
<tr>
<td>NLPCTI Format</td>
<td>233</td>
</tr>
<tr>
<td>NLPCTN Format</td>
<td>235</td>
</tr>
<tr>
<td>NLPCTP Format</td>
<td>236</td>
</tr>
<tr>
<td>NLPVALUE Format</td>
<td>237</td>
</tr>
<tr>
<td>NLSRMON Format</td>
<td>238</td>
</tr>
<tr>
<td>NLSRQTR Format</td>
<td>239</td>
</tr>
<tr>
<td>NLSRWK Format</td>
<td>240</td>
</tr>
<tr>
<td>NLTIMAP Format</td>
<td>241</td>
</tr>
<tr>
<td>NLTIME Format</td>
<td>242</td>
</tr>
</tbody>
</table>
Categories of NLS Formats

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>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>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>Category</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</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>Character</td>
<td>SEBCDIC Format (p. 92)</td>
<td>Converts native format character data to EBCDIC representation.</td>
</tr>
<tr>
<td></td>
<td>$UCS2B Format (p. 243)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS2BE Format (p. 245)</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>$UCS2L Format (p. 246)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS2LE 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></td>
<td>$UCS2X Format (p. 248)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS2XE Format (p. 250)</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>
</tr>
<tr>
<td></td>
<td>$UCS4B Format (p. 251)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 32-bit, UCS4, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS4BE Format (p. 252)</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>
</tr>
<tr>
<td></td>
<td>$UCS4L Format (p. 253)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 32-bit, UCS4, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS4LE Format (p. 255)</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>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Category</strong></td>
<td><strong>Language Elements</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Language Elements</strong></td>
<td><strong>Description</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>$UCS4X$ Format (p. 256)</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. 257)</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. 258)</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. 264)</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>$UTF8X$ Format (p. 265)</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. 266)</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><strong>Date and Time</strong></td>
<td><strong>B8601DX</strong> Format (p. 84)</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><strong>B8601LX</strong> Format (p. 86)</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+</td>
<td>-hhmm.</td>
</tr>
<tr>
<td><strong>B8601TX</strong> Format (p. 87)</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 hhmms+</td>
<td>-hhmm.</td>
</tr>
<tr>
<td><strong>E8601DX</strong> Format (p. 93)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time</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></td>
<td>by using the ISO 8601 datetime and time zone extended notation yyyy-mm-ddThh:mm:ss+</td>
</tr>
<tr>
<td>E8601LX Format (p. 94)</td>
<td></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>E8601TX Format (p. 96)</td>
<td></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>MINGUO Format (p. 102)</td>
<td></td>
<td>Writes date values as Taiwanese dates in the form yyyyymmdd.</td>
</tr>
<tr>
<td>NENGO Format (p. 104)</td>
<td></td>
<td>Writes date values as Japanese dates in the form e.yymmdd.</td>
</tr>
<tr>
<td>NLDATE Format (p. 107)</td>
<td></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>NLDATEL Format (p. 108)</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, date, month, year.</td>
</tr>
<tr>
<td>NLDATEM Format (p. 109)</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.</td>
</tr>
<tr>
<td>NLDATEMD Format (p. 110)</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.</td>
</tr>
<tr>
<td>NLDATEMDL Format (p. 111)</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.</td>
</tr>
<tr>
<td>NLDATEMDM Format (p. 111)</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.</td>
</tr>
<tr>
<td>NLDATEMDS Format (p. 112)</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.</td>
</tr>
<tr>
<td>NLDATEN Format (p. 113)</td>
<td></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>NLDATES Format (p. 114)</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 string.</td>
</tr>
<tr>
<td>NLDATEG Format (p. 115)</td>
<td></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>NLDATENW Format (p. 116)</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 day of the week.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLDATEYM Format (p. 118)</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>
<td></td>
</tr>
<tr>
<td>NLDATEYML Format (p. 119)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the month and year.</td>
<td></td>
</tr>
<tr>
<td>NLDATEYMM Format (p. 120)</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>
<td></td>
</tr>
<tr>
<td>NLDATEYMS Format (p. 121)</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>
<td></td>
</tr>
<tr>
<td>NLDATEYQ Format (p. 122)</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>
<td></td>
</tr>
<tr>
<td>NLDATEYQL 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>
<td></td>
</tr>
<tr>
<td>NLDATEYQM 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 (Q1–Q4) using abbreviations.</td>
<td></td>
</tr>
<tr>
<td>NLDATEYQS Format (p. 125)</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>
<td></td>
</tr>
<tr>
<td>NLDATEYR 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.</td>
<td></td>
</tr>
<tr>
<td>NLDATEYW Format (p. 127)</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>
<td></td>
</tr>
<tr>
<td>NLDATM Format (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.</td>
<td></td>
</tr>
<tr>
<td>NLDATMAP Format (p. 129)</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>
<td></td>
</tr>
<tr>
<td>NLDATMDT Format (p. 130)</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>
<td></td>
</tr>
<tr>
<td>NLDATML Format (p. 131)</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>
<td></td>
</tr>
<tr>
<td>NLDATMM Format (p. 132)</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>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------------------- Adam's answer is correct.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLDATMYQ Format (p. 146)</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. 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 value (1–4) and the year.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQM 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’s quarter (1–4) and then the year.</td>
<td></td>
</tr>
<tr>
<td>NLDATMYQS Format (p. 149)</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. 150)</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. 151)</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. 152)</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>WEEKU Format (p. 267)</td>
<td>Writes a week number in decimal format by using the U algorithm.</td>
<td></td>
</tr>
<tr>
<td>WEEKV Format (p. 268)</td>
<td>Writes a week number in decimal format by using the V algorithm.</td>
<td></td>
</tr>
<tr>
<td>WEEKW Format (p. 270)</td>
<td>Writes a week number in decimal format by using the W algorithm.</td>
<td></td>
</tr>
<tr>
<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>
<td></td>
</tr>
<tr>
<td>YYWEEKV Format (p. 274)</td>
<td>Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.</td>
<td></td>
</tr>
<tr>
<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>
<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>Hebrew Text Handling</td>
<td>$\text{SCPTDW Format}$ (p. 90)</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>$\text{SCPTWD Format}$ (p. 91)</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. 84)</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 basic notation $\text{yyyyymmddThhmmss+hhmm}$.</td>
</tr>
<tr>
<td></td>
<td>B8601LX Format (p. 86)</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 $\text{yyyyymmddThhmmss+</td>
</tr>
<tr>
<td></td>
<td>B8601TX Format (p. 87)</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 $\text{hhmmss+</td>
</tr>
<tr>
<td></td>
<td>E8601DX Format (p. 93)</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 extended notation $\text{yyyy-mm-ddThh:mm:ss+hh:mm}$.</td>
</tr>
<tr>
<td></td>
<td>E8601LX Format (p. 94)</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 $\text{yyyy-mm-ddThh:mm:ss+</td>
</tr>
<tr>
<td></td>
<td>E8601TX Format (p. 96)</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 $\text{hh:mm:ss+</td>
</tr>
<tr>
<td>Numeric</td>
<td>BESTDOTX Format (p. 88)</td>
<td>Specifies that SAS choose the best notation and use a dot as a decimal separator.</td>
</tr>
<tr>
<td></td>
<td>EURO Format (p. 97)</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. 100)</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>NLBLBEST Format (p. 105)</td>
<td>Writes the best numerical notation based on the locale.</td>
</tr>
<tr>
<td></td>
<td>NLMNIAED Format (p. 153)</td>
<td>Writes the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>NLMNIAUD Format (p. 154)</td>
<td>Writes the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIBGN Format (p. 155)</td>
<td>Writes the monetary format of the international expression for Bulgaria.</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>NLMNIBRL Format (p. 156)</td>
<td>Writes the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNICAD Format (p. 157)</td>
<td>Writes the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNICHF Format (p. 158)</td>
<td>Writes the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNICNY Format (p. 159)</td>
<td>Writes the monetary format of the international expression for China.</td>
</tr>
<tr>
<td></td>
<td>NLMNICZK Format (p. 160)</td>
<td>Writes the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>NLMNIDKK Format (p. 161)</td>
<td>Writes the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>NLMNIEEEK Format (p. 162)</td>
<td>Writes the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIEGP Format (p. 163)</td>
<td>Writes the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>NLMNIEEUR Format (p. 164)</td>
<td>Writes the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td></td>
<td>NLMNIGBP Format (p. 165)</td>
<td>Writes the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td>NLMNIHKD Format (p. 166)</td>
<td>Writes the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td></td>
<td>NLMNIHRK Format (p. 167)</td>
<td>Writes the monetary format of the international expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIHUF Format (p. 168)</td>
<td>Writes the monetary format of the international expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>NLMIIIDR Format (p. 169)</td>
<td>Writes the monetary format of the international expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIILS Format (p. 170)</td>
<td>Writes the monetary format of the international expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNIINR Format (p. 171)</td>
<td>Writes the monetary format of the international expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNIJPY Format (p. 172)</td>
<td>Writes the monetary format of the international 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>NLMNIKRW Format</td>
<td>(p. 173)</td>
<td>Writes the monetary format of the international expression for South Korea.</td>
</tr>
<tr>
<td>NLMNILTL Format</td>
<td>(p. 174)</td>
<td>Writes the monetary format of the international expression for Lithuania.</td>
</tr>
<tr>
<td>NLMNILVL Format</td>
<td>(p. 175)</td>
<td>Writes the monetary format of the international expression for Latvia.</td>
</tr>
<tr>
<td>NLMNIMOP Format</td>
<td>(p. 176)</td>
<td>Writes the monetary format of the international expression for Macau.</td>
</tr>
<tr>
<td>NLMNIMXN Format</td>
<td>(p. 177)</td>
<td>Writes the monetary format of the international expression for Mexico.</td>
</tr>
<tr>
<td>NLMNIMYR Format</td>
<td>(p. 178)</td>
<td>Writes the monetary format of the international expression for Malaysia.</td>
</tr>
<tr>
<td>NLMNINOK Format</td>
<td>(p. 179)</td>
<td>Writes the monetary format of the international expression for Norway.</td>
</tr>
<tr>
<td>NLMNINZD Format</td>
<td>(p. 180)</td>
<td>Writes the monetary format of the international expression for New Zealand.</td>
</tr>
<tr>
<td>NLMNIPN Format</td>
<td>(p. 181)</td>
<td>Writes the monetary format of the international expression for Poland.</td>
</tr>
<tr>
<td>NLMNIRUB Format</td>
<td>(p. 182)</td>
<td>Writes the monetary format of the international expression for Russia.</td>
</tr>
<tr>
<td>NLMNISEK Format</td>
<td>(p. 183)</td>
<td>Writes the monetary format of the international expression for Sweden.</td>
</tr>
<tr>
<td>NLMNISGD Format</td>
<td>(p. 184)</td>
<td>Writes the monetary format of the international expression for Singapore.</td>
</tr>
<tr>
<td>NLMNITHB Format</td>
<td>(p. 185)</td>
<td>Writes the monetary format of the international expression for Thailand.</td>
</tr>
<tr>
<td>NLMNITRY Format</td>
<td>(p. 186)</td>
<td>Writes the monetary format of the international expression for Turkey.</td>
</tr>
<tr>
<td>NLMNITWD Format</td>
<td>(p. 187)</td>
<td>Writes the monetary format of the international expression for Taiwan.</td>
</tr>
<tr>
<td>NLMNIUSD Format</td>
<td>(p. 188)</td>
<td>Writes the monetary format of the international expression for Puerto Rico and the United States.</td>
</tr>
<tr>
<td>NLMNIZAR Format</td>
<td>(p. 189)</td>
<td>Writes the monetary format of the international expression for South Africa.</td>
</tr>
<tr>
<td>NLMNLAED Format</td>
<td>(p. 190)</td>
<td>Writes the monetary format of the local expression for the United Arab Emirates.</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>NLMNLAUD Format (p. 191)</td>
<td>Writes the monetary format of the local expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBGN Format (p. 192)</td>
<td>Writes the monetary format of the local expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBRL Format (p. 193)</td>
<td>Writes the monetary format of the local expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCAD Format (p. 194)</td>
<td>Writes the monetary format of the local expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCHF Format (p. 195)</td>
<td>Writes the monetary format of the local expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCNY Format (p. 196)</td>
<td>Writes the monetary format of the local expression for China.</td>
</tr>
<tr>
<td></td>
<td>NLMNLČZK Format (p. 197)</td>
<td>Writes the monetary format of the local expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>NLMNLDKK Format (p. 198)</td>
<td>Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEEK Format (p. 199)</td>
<td>Writes the monetary format of the local expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEG Format (p. 200)</td>
<td>Writes the monetary format of the local expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>NLMNLEUR Format (p. 201)</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>
</tr>
<tr>
<td></td>
<td>NLMNLGBP Format (p. 202)</td>
<td>Writes the monetary format of the local expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHKD Format (p. 203)</td>
<td>Writes the monetary format of the local expression for Hong Kong.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHRK Format (p. 204)</td>
<td>Writes the monetary format of the local expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLHUF Format (p. 205)</td>
<td>Writes the monetary format of the local expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>NLMNLDIDR Format (p. 206)</td>
<td>Writes the monetary format of the local expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLILS Format (p. 207)</td>
<td>Writes the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNLINR Format (p. 208)</td>
<td>Writes the monetary format of the local expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNLJPY Format (p. 209)</td>
<td>Writes the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLKRW Format (p. 210)</td>
<td>Writes the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLTL Format (p. 211)</td>
<td>Writes the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLVL Format (p. 212)</td>
<td>Writes the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLMNLMOP Format (p. 213)</td>
<td>Writes the monetary format of the local expression for Macau.</td>
<td></td>
</tr>
<tr>
<td>NLMNLMXN Format (p. 214)</td>
<td>Writes the monetary format of the local expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>NLMNLMYR Format (p. 215)</td>
<td>Writes the monetary format of the local expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNOK Format (p. 216)</td>
<td>Writes the monetary format of the local expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>NLMLNZNZD Format (p. 217)</td>
<td>Writes the monetary format of the local expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>NLMNLPLN Format (p. 218)</td>
<td>Writes the monetary format of the local expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLRUB Format (p. 219)</td>
<td>Writes the monetary format of the local expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLSEK Format (p. 220)</td>
<td>Writes the monetary format of the local expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>NLMNLSGD Format (p. 221)</td>
<td>Writes the monetary format of the local expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTTHB Format (p. 222)</td>
<td>Writes the monetary format of the local expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTTRY Format (p. 223)</td>
<td>Writes the monetary format of the local expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTWD Format (p. 224)</td>
<td>Writes the monetary format of the local expression for Taiwan.</td>
<td></td>
</tr>
<tr>
<td>NLMNLUSD Format (p. 225)</td>
<td>Writes the monetary format of the local expression for Puerto Rico and the United States.</td>
<td></td>
</tr>
<tr>
<td>NLMNLZAR Format (p. 226)</td>
<td>Writes the monetary format of the local expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>NLMNY Format (p. 227)</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. 228)</td>
<td>Writes the monetary format of the international expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLNUM Format (p. 229)</td>
<td>Writes the numeric format of the local expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLNUMI Format (p. 231)</td>
<td>Writes the numeric format of the international expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLPCT Format (p. 232)</td>
<td>Writes percentage data of the local expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLPCTI Format (p. 233)</td>
<td>Writes percentage data of the international expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLPCTN Format (p. 235)</td>
<td>Produces percentages, using a minus sign for negative values.</td>
<td></td>
</tr>
<tr>
<td>NLPCTP Format (p. 236)</td>
<td>Writes locale-specific numeric values as percentages.</td>
<td></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 `yyyyymmddThhmmss` +`hhmm`.

**Categories:** Date and Time  
ISO 8601  
**Alignment:** Left  
**Supports:** ISO 8601 Element 5.4.1, complete representation

### Syntax

`B8601DXw`.

### Syntax Description

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

  **Default:** 26  
  **Range:** 20–35

### Details

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

## NLP VALUE Format (p. 237)

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

## NLSTRMON Format (p. 238)

Writes the month name in the specified locale.

## NLSTRQTR Format (p. 239)

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

## NLSTRWK Format (p. 240)

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

## YEN Format (p. 271)

Writes numeric values with yen signs, commas, and decimal points.
on the local date and time. The B8601DX format writes SAS datetime values by using the following ISO 8601 basic datetime notation:

- yyyymmddThhmmss+hhmm

**yyy**
- is a four-digit year.

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

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

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

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

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

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

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

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

**Example**

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

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

- yyyymmddThhmss+|-hhmm

  yyy  
is a four-digit year.

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

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

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

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

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

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

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

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

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

```plaintext
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:
•  `hhmmss [+–]hhmm`

`hh`

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

`mm`

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

`ss`

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

`+[–]hhmm`

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

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

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

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

**Example**

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

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

**See Also**

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

**BESTDOTX Format**

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

**Category:** Numeric
Syntax

BESTDOTXw.

Syntax Description

w

specifies the width of the output field.

Default 12

Range 1–32

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

Details

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

<table>
<thead>
<tr>
<th>LOCALE option</th>
<th>Default decimal separator character for the locale</th>
<th>NLDECSEPARATOR option</th>
<th>Separator character used by 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 NLSDECSEPARATOR option enabled and the locale set to fr_FR.

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

\begin{tabular}{|l|l|}
\hline
\text{&put \&macx;} & 1.2 \\
\hline
\text{data _null_;} & 1,2 \\
\text{bx=&macx; /* Succeeds because the value of \&macx is 1.2 */} & \\
\text{put bx;} & \\
\text{run;} & \\
\hline
\end{tabular}

\textbf{See Also}

- “BEST Format” in \textit{SAS Viya Formats and Informats: Reference}
- “NLDECSEPARATOR System Option” on page 556

\textbf{$\textsc{Cptdw}$ 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).

\begin{tabular}{|l|}
\hline
\textbf{Category:} & Hebrew Text Handling \\
\hline
\textbf{Alignment:} & Left \\
\hline
\end{tabular}

\textbf{Syntax}

\textsc{Cptdw}w.

\textbf{Syntax Description}

\begin{itemize}
\item \textit{w} specifies the width of the output field.
\end{itemize}

\begin{tabular}{|l|l|}
\hline
\textbf{Default} & 200 \\
\hline
\textbf{Range} & 1–32767 \\
\hline
\end{tabular}

\textbf{Comparisons}

The $\textsc{Cptdw}\_w$. format performs processing that is the opposite of the $\textsc{Cptwd}\_w$. format.

\textbf{Example}

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

\begin{tabular}{|l|l|}
\hline
\textbf{Statement} & \textbf{Result} \\
\hline
\text{put text \$cptdw3.;} & 118 \\
\hline
\end{tabular}
$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**

$CPTWDw.

**Syntax Description**

\(w\)

specifies the width of the output field.  

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1–32767</td>
</tr>
</tbody>
</table>

**Comparisons**

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

**Example**

The following example uses the input value of “לאריה”.

<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**

**Format:**
$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

`$EBCDICw`.

### 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 19.2 on page 606.

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 555.

### 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.;
```
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

**Alignment:**
- Left

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

### Syntax

**E8601DX**

### Syntax Description

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

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>20–35</td>
</tr>
</tbody>
</table>

### Details

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

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

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

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

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

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

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

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

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

Example

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

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
</table>
| data _null_;
t='01Feb2013T12:34:56'dt;
put t e8601dx.;
run;          | 2013-02-01T07:34:56-05:00 |
| options timezone='America/Adak';
data _null_;
t='01Feb2013T12:34:56'dt;
put t e8601dx.;
run;          | 2013-02-01T02:34:56-10:00 |

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

E8601LX

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;
put blx e8601lx.;
Value of blx | Result
167541296    | 2013-02-01T12:34:56-05:00

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

Syntax
E8601TXw.

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+|–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 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.

- **Category:** Numeric
- **Alignment:** Right

**Syntax**

```
EUROw.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$w.d$ format is similar to the EUROX$w.d$ format, but EUROX$w.d$ format reverses the roles of the decimal point and the comma. This convention is common in European countries.

• The EURO$w.d$ format is similar to the DOLLAR$w.d$ format, except that DOLLAR$w.d$ 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></td>
<td>----+----1----+----2----+----3</td>
</tr>
<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  
;  
rung;  
SAS Log:  
  E1,254.71  
  E1255
/* 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;
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.

**Category:** Numeric

**Alignment:** Right

### Syntax

$\text{EUROX} w.d$

#### 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 EUROXw,d format is similar to the EUROw,d format, but EUROw,d format reverses the roles of the comma and the decimal point. This convention is common in English-speaking countries.

- The EUROXw,d format is similar to the DOLLARXw,d format, except that DOLLARXw,d format writes a leading dollar sign instead of the euro symbol.

Example

These examples use 1254.71 as the value of amount.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put amount eurox10.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox5.;</td>
<td>E1255</td>
</tr>
<tr>
<td>put amount eurox9.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox15.3;</td>
<td>E1.254,710</td>
</tr>
</tbody>
</table>

```sas
data _null_;  
    input x;  
    put x eurox10.2;  
    put x eurox5.;  
    put x eurox9.2;  
    put x eurox15.3;  
    datalines;  
1254.71  
; run;  
SAS Log:  
  E1.254,71  
  E1255  
  E1.254,71  
    E1.254,710  
/* This code determines the default length. */
data _null_;  
    input x;  
    put x eurox.;  
    datalines;  
1  
22  
333  
4444  
55555  
666666  
7777777  
88888888  
999999999  
1234561234  
```
Note: At least one W.D format was too small for the number to be printed. The decimal might be shifted by the "BEST" format.

See Also

Format:

- “EURO Format” on page 97

Informats:

- “EURO Informat” on page 416
- “EUROX Informat” on page 418

MINGUO Format

Writes date values as Taiwanese dates in the form yyyymmdd.

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 yyyymmdd, where

yyyy
is an integer that represents the year.
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>\texttt{x=put(12054,minguo7.); put x=;}</td>
<td>\texttt{x=820101}</td>
</tr>
<tr>
<td>\texttt{x=put(12054,minguo9.); put x=;}</td>
<td>\texttt{x=82/01/01}</td>
</tr>
<tr>
<td>\texttt{x=put(12054,minguo10.); put x=;}</td>
<td>\texttt{x=0082/01/01}</td>
</tr>
<tr>
<td>\texttt{x=put(18993,minguo7.); put x=;}</td>
<td>\texttt{x=1000101}</td>
</tr>
<tr>
<td>\texttt{x=put(18993,minguo9.); put x=;}</td>
<td>\texttt{x=100/01/01}</td>
</tr>
<tr>
<td>\texttt{x=put(18993,minguo10.); put x=;}</td>
<td>\texttt{x=0101/01/01}</td>
</tr>
<tr>
<td>\texttt{x=put(-20088,minguo7.); put x=;}</td>
<td>\texttt{*******}</td>
</tr>
<tr>
<td>\texttt{x=put(-20088,minguo9.); put x=;}</td>
<td>\texttt{*******}</td>
</tr>
<tr>
<td>\texttt{x=put(-20088,minguo10.); put x=;}</td>
<td>\texttt{*******}</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “MINGUO Informat” on page 419
NENGO Format

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

Category: Date and Time
Alignment: Left

Syntax

NENGOw;

Syntax Description

w
specifies the width of the output field.

Default 10
Range 2–10

Details

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

data _null_;  
  date=15342;    
  put date nengo3.;  
  put date nengo6.;  
  put date nengo8.;  
  put date nengo9.;  
  put date nengo10.;    
run
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date nengo1.;</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:**
- “NENGO Informat” on page 421

### NLBEST Format

Writers the best numerical notation based on the locale.

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

### Syntax

**Syntax**

\begin{verbatim}
NLBEST\w.
\end{verbatim}

### 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>1–32</td>
</tr>
</tbody>
</table>

**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:

```plaintext
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>
<tr>
<td>locale=German_Germany</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>
<tr>
<td>Locales</td>
<td>Results</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>locale=ar_BH</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>

**NLDATE Format**

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

**Category:** Date and Time

**Alignment:** Left

**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**

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

**Example**

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<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 113
- “NLDATEW Format” on page 115
- “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.

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

**Syntax**

NLDATEL\(w\).

**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 outputs 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.
Statements | Results
--- | ---
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
run;

--- 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

**NLDATEM Format**

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

- **Category:** Date and Time
- **Alignment:** Left

**Syntax**

NLDATEM<sub>w</sub>

**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>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>2–200</td>
</tr>
</tbody>
</table>

**Details**

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

**Example**

This example uses the date November 19, 2012.
### 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.

**Category:** Date and Time  
**Alignment:** Left

#### Syntax

\[
\text{NLDATEMD}_w.
\]

#### 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 118
**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.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

```
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 outputs 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.

```
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATEMDL min=5 default=12 max=200 ----+";  
put dy nldatemdl.;  
put dy nldatemdl5.;  
put dy nldatemdl15.;  
put dy nldatemdl19.;  
put dy nldatemdl12.;  
put dy nldatemdl200.;  
run;
```

**Results**

+--- NLDATEMDL min=5 default=12 max=200 ----+  
November 19  
11/19  
Nov 19  
November 19  
November 19

---

**NLDATEEMDM 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**

NLDATEMDM<sub>w</sub>.

**Syntax Description**

<sub>W</sub>

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 outputs 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.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATEMDS<sub>w</sub>.  

---
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 outputs 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>put dy nldatemds.;</td>
<td>11/19</td>
</tr>
<tr>
<td>put dy nldatemds5.;</td>
<td>11/19</td>
</tr>
<tr>
<td>put dy nldatemds5.;</td>
<td>11/19</td>
</tr>
<tr>
<td>put dy nldatemds200.;</td>
<td>11/19</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.

Category: Date and Time

Alignment: Left

Syntax

\[ \text{NLDATEMN}_w. \]

Syntax Description

\( w \)

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

Default: 9
Range 4–200

**Comparisons**

NLDATEMN\(w\). is similar to MONNAME\(w\). except that NLDATEMN\(w\). 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>options locale=English_UnitedStates;</td>
<td>February</td>
</tr>
<tr>
<td>put month nldatemn.;</td>
<td>February</td>
</tr>
</tbody>
</table>

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

**See Also**

**Formats:**

- “NLDATE Format” on page 107
- “NLDATEW Format” on page 115
- “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.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATES\(w\).

**Syntax Description**

\(w\)

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

NLDATES outputs 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 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>11/19/2012</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>11/19/2012</td>
</tr>
<tr>
<td>put dy nldates10.;</td>
<td>11/19/2012</td>
</tr>
<tr>
<td>put dy nldates200.;</td>
<td>11/19/2012</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.

- **Category:** Date and Time
- **Alignment:** Left

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 NLDATETIMEW 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 67 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Comparisons

NLDATETIMEWw. is similar to WEEKDATEW. except that NLDATETIMEWw. 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;</td>
<td>Monday, February 24, 2003</td>
</tr>
<tr>
<td>date=15760;</td>
<td>Mon, Feb 24, 2003</td>
</tr>
<tr>
<td>x=put(date,nldatetimew.);</td>
<td>Monday, February 24, 2003</td>
</tr>
<tr>
<td>y=put(date,nldatetimew20.);</td>
<td></td>
</tr>
<tr>
<td>z=put(date,nldatetimew200.);</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

| options locale=German_Germany; | Mo., 24. Feb 2003 |
| date=15760; | Mo., 24. Feb 2003 |
| x=put(date,nldatetimew.); | |
| y=put(date,nldatetimew20.); | Montag, 24. Februar 2003 |
| z=put(date,nldatetimew200.); | |
| run; | |

See Also

Formats:

- “NLDATETIME Format” on page 107
- “NLDATETIMEMN Format” on page 113
- “NLDATETIMEWN Format” on page 116

NLDATETIMEWN 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.
Syntax

NLDATEWNw.

Syntax Description

w

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

Default: 9
Range: 4–200

Comparisons

NLDATEWNw. is similar to DOWNAMEw. except that NLDATEWNw. is locale-specific.

Example

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>Monday</td>
</tr>
<tr>
<td>put date nldatewn.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>Montag</td>
</tr>
<tr>
<td>put date nldatewn.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- “NLDATE Format” on page 107
- “NLDATEMN Format” on page 113
- “NLDATEW Format” on page 115
**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.

- **Category:** Date and Time
- **Alignment:** Left

**Syntax**

\[ \text{NLDATEYM}_w. \]

**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 output the date without the asterisks:

```
PROC LOCALEDATA;
  LOAD SASLOCALE;
  MODIFY key=DATE_YYMM_SHORT_FORMAT value='%b %y' ;
  SAVE REGISTRY / _ALL_ syntax=SAS;
```

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

\[ d=\text{Oct 14} \]

**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></td>
<td></td>
</tr>
</tbody>
</table>
options locale=spanish_Spain;
data_null_;
dy=today();
x=put(dy, nldateym.);
y=put(dy, nldateym12.);
z=put(dy, nldateym200.);
runcase of
See Also

Format:
• “NLDATEMD Format” on page 110

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

Category: Date and Time
Alignment: Left

Syntax
NLDATEMYMLw.

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
NLDATEMYML outputs 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.
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.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

NLDATEYMMw.

**Syntax Description**

w  

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

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>5–200</td>
</tr>
</tbody>
</table>

**Details**

NLDATEYMS outputs 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.
Statements

```sas
data _null_;
  dt = datetime();
  dy = date();
  put "+--- NLDATEYMM min=5 default=11 max=200 ---+";
  put dy nldateymm.;
  put dy nldateymm7.;
  put dy nldateymm11.;
  put dy nldateymm200.;
run;
```

Results

```
+--- NLDATEYMM min=5 default=11 max=200 ---+
Nov 2012
11/2012
Nov 2012
Nov 2012
```

---

### 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.

**Category:** Date and Time  
**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 outputs the date in a short-uniform pattern with numbers and delimiters such as mm/yyyy.

---

### Example

This example uses the date November 19, 2012.
Statements | Results
---|---
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATEYMS min=5 default=7 max=200 ---+
11/2012  
11/12  
11/2012  
11/2012  
run;

**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.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

NLDATEYQ\textit{w}.

**Syntax Description**

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

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4–200</td>
</tr>
</tbody>
</table>

**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 67 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.
### NLDATEYQL 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.

- **Category:** Date and Time
- **Alignment:** Left

#### Syntax

```
NLDATEYQL\ 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>18</td>
<td>4–200</td>
</tr>
</tbody>
</table>

#### Details

NLDATEYQL outputs 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.
### Statements

```sas
data _null_
  dt = datetime()
  dy = date()
  put "+--- NLDATEYQL min=4 default=18 max=200 ---+
      4th quarter 2013
      2013/4
      Q4 2013
      4th quarter 2013
    
    put dy nldateyql.
    put dy nldateyql6..;
    put dy nldateyql7..;
    put dy nldateyql18..;
    put dy nldateyql200..;
  run;
```

### Results

```
+--- NLDATEYQL min=4 default=18 max=200 ---+
      4th quarter 2013
      2013/4
      Q4 2013
      4th quarter 2013
    
      4th quarter 2013
      4th quarter 2013
    
    4th quarter 2013
    4th quarter 2013
```

### 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.

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

### 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 outputs 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.
## 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.

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

### Syntax

\[
\text{NLDATEYQS}w. 
\]

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

### Details

NLDATEYQS outputs 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.

```plaintext
data _null_;  
dt = datetime();  
dy = date();  
put ’---- NLDATEYQM min=4 default=7 max=200 ----’;  
  put dy nldateyqm.;  
  put dy nldateyqm6.;  
  put dy nldateyqm7.;  
  put dy nldateyqm200.;  
run;
```
## NLDATEYR Format

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

**Category:** Date and Time  
**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.
Statements

options locale=fr_FR;

data _null_
;
dy=today();
dt=datetime();
put "+--- NLDATEYR min=2 default=16 max=200 ---+

NLDATEYR 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.

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

Syntax

NLDATEYWw.

Syntax Description

w

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>5–200</td>
</tr>
</tbody>
</table>

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.

**Category:** Date and Time  
**Alignment:** Left

#### Syntax

NLDATMw.

#### Syntax Description

w  
Specifies the width of the output field. If necessary, SAS abbreviates the datetime value to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10–200</td>
</tr>
</tbody>
</table>

#### 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 | Results
---|---
| ----|----|----
| 1 | 2 | 3
options locale=English_UnitedStates; 24Feb2003:12:39:43
put day nldatm.;

options locale=German_Germany; 24. Februar 2003 12.39 Uhr
put day nldatm.;

See Also

Formats:

- “NLDATMAP Format” on page 129
- “NLDATMTM Format” on page 138
- “NLDATMW Format” on page 140

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.

Category: Date and Time
Alignment: Left

Syntax

NLDATMAPw.

Syntax Description

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

Default 32
Range 16–200

Comparisons

The NLDATMAPw. format is similar to DATEAMPMw. except that the NLDATMAPw. format is locale-specific.

Example

These examples 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.
See Also

Formats:

- “NLDATM Format” on page 128
- “NLDATMTM Format” on page 138
- “NLDATMW Format” on page 140

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.

- **Category:** Date and Time
- **Alignment:** Left

**Syntax**

\[ \text{NLDATMDT}_w \]

**Syntax Description**

- **w**
  - specifies the width of the output field
  - **Default:** 18
  - **Range:** 10-200

**Example**

This example uses the en_US locale option.
NLDATML 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.

Syntax

\[ \text{NLDATML}_w \]

Syntax Description

\( w \)

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

Default 30

Range 9–200

Details

NLDATML outputs 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

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

Results

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATML min=9 default=30 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>dy = date();</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>put &quot;+--- NLDATML min=9 default=30 max=200 ---+&quot;;</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>put dt nldatml.;</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>put dt nldatml19.;</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>put dt nldatml24.;</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>put dt nldatml30.;</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>put dt nldatml200.;</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

### 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.

**Category:** Date and Time  
**Alignment:** Left

#### Syntax

```sas
NLDATMMw.
```

#### Syntax Description

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

**Default:** 24  
**Range:** 9–200

#### Details

NLDATMM outputs 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.

**Category:** Date and Time  
**Alignment:** Left
Syntax

NLDATMMD\(w\).

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>options locale=en_US;</td>
<td>January 02</td>
</tr>
<tr>
<td>data <em>null</em> x=put(86400,nldatmmd.); put x=; run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLDATMYM Format” on page 143

---

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.

Category: Date and Time

Alignment: Left

Syntax

NLDATMMDL\(w\).

Syntax Description

\(w\)

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

Default 9
Details

NLDATMMDL outputs 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.

```sas
data _null_;
  dt = datetime();
  dy = date();
  put "+--- NLDATMMDL min=5 default=12 max=200 ---+";
  put dt nldatmmdl.;
  put dt nldatmmdl5.;
  put dt nldatmmdl9.;
  put dt nldatmmdl12.;
  put dt nldatmmdl200.;
run;
```

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.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATMMDM<sup>W</sup>.

**Syntax Description**

<sup>W</sup>

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

**Default** 9

**Range** 5–200

Details

NLDATMMDM outputs 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_;  +--- NLDATMMDM min=5 default=9 max=200 ---+
  dt = datetime();                Nov 19
  dy = date();                    11/19
  put "++++ NLDATMMDM min=5 default=9 Nov 19
      max=200 ++++";             Nov 19
  put dt nldatmmdm.;             Nov 19
  put dt nldatmmdm5.;            Nov 19
  put dt nldatmmdm9.;            Nov 19
  put dt nldatmmdm200.;          Nov 19
run;
```

### 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.

- **Category:** Date and Time
- **Alignment:** Left

### Syntax

```
NLDATMMDSw.
```

### 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 outputs 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.
Statements                                      Results

data _null_;                                    +++ NLDATMMDS min=5 default=5 max=200 +++
  dt = datetime();                               11/19
  dy = date();                                   11/19
  put "+++ NLDATMMDS min=5 default=5 max=200 +++
      ";                                        11/19
  put dt nldatmmds.;                             11/19
  put dt nldatmmds5.;                            11/19
  put dt nldatmmds5.;                            11/19
  put dt nldatmmds200.;                          11/19
run;

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.

Category:          Date and Time
Alignment:         Left

Syntax

NLDATMMNw.

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.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

```
NLDATMS w.
```

**Syntax Description**

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

  **Default** 19  
  **Range** 9–200

**Details**

NLDATMS outputs 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.

---

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

<table>
<thead>
<tr>
<th>data <em>null</em>;</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt = datetime();</td>
<td>+--- NLDATMS min=9 default=19 max=200 ---+</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19/2012 14:38:52</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMS min=9 default=19 max=200 ---+&quot;;</td>
<td>11/19/2012 14:38:52</td>
</tr>
<tr>
<td>put dt nldatms.;</td>
<td>11/19/2012 14:38:52</td>
</tr>
<tr>
<td>put dt nldatms10.;</td>
<td>11/19/2012 14:38:52</td>
</tr>
<tr>
<td>put dt nldatms19.;</td>
<td>11/19/2012 14:38:52</td>
</tr>
<tr>
<td>put dt nldatms200.;</td>
<td>11/19/2012 14:38:52</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

---

### NLDATMTM Format

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

**Category:** Date and Time  
**Alignment:** Left

### Syntax

NLDATMTM<sub>w</sub>. 

### Syntax Description

<sub>w</sub>  
specifies the width of the output field.  

- **Default:** 16  
- **Range:** 16–200

### Comparisons

The NLDATMTM<sub>w</sub> format is similar to the TOD<sub>w</sub> format except that the NLDATMTM<sub>w</sub> 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>options locale=English_UnitedStates;</td>
<td>12:39:43</td>
</tr>
<tr>
<td>put event nldatmtm.;</td>
<td></td>
</tr>
</tbody>
</table>
### NLDATMTZ Format

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

**Category:** Date and Time  
**Alignment:** Left

#### Syntax

``` SAS 
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>
</table>
| options locale=fr_FR;  
data test;  
x=datetime();  
put x=nldatmtz.;  
run; | x=10 h 40 -0400 |

---

### See Also

**Formats:**

- “NLDATM Format” on page 128  
- “NLDATMAP Format” on page 129  
- “NLDATMW Format” on page 140
NLDATMW Format

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

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

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 67 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\(w\) format is similar to the TWMDY\(w\) format except that the NLDATMW\(w\) 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>

---+---1---+---2---+---3
### Statements

```sas
options locale=English_UnitedStates;

data _null_;  
x=put(1361709583,nldatmw.);  
y=put(1361709583,nldatmw30.);  
z=put(1361709583,nldatmw200.);  
run;
```

### Results

```text
Mon, Feb 24, 2003 12:39:43 PM

Monday, February 24, 2003 12:39:43 PM
```

### Statements

```sas
options locale=german_germany;

data _null_;  
x=put(1361709583,nldatmw.);  
y=put(1361709583,nldatmw30.);  
z=put(1361709583,nldatmw200.);  
run;
```

### Results

```text
Mo, 24. Feb 2003 12.39 Uhr

Montag, 24. Februar 2003 12.39 Uhr
```

---

### See Also

**Formats:**

- “NLDATM Format” on page 128
- “NLDATMAP Format” on page 129
- “NLDATMTM Format” on page 138

---

### 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.

**Category:** Date and Time  
**Alignment:** Left

---

### Syntax

\texttt{NLDATMWN} \texttt{w}.

### Syntax Description

\texttt{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>+--- NLDATMWZ 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;+--- NLDATMWZ min=4 default=9</td>
<td>Monday</td>
</tr>
<tr>
<td>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></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.

- **Category:** Date and Time
- **Alignment:** Left

Syntax

NLDATMWZ<sub>w</sub>.

Syntax Description

<sub>w</sub>

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 67 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
--- | ---
```options locale=fr_FR;``` | ```x=vendredi 18 mars 2011 10 h 40 -0400```
```data test;``` | ```x=datetime();```
```put x=nldatmwz.;``` | ```run;```

---

**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.

**Category:** 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.

Statement | Result
--- | ---
```options locale=en_US;``` | January 1960
```data _null_;``` | January 1960
```x=put(86400,nldatym.);``` | `x=put(86400,nldatym12.);`
```put x=;``` | ```put y=;```
```run;``` | ```run;```

---

**See Also**

**Format:**
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.

Category: Date and Time
Alignment: Left

Syntax

NLDATMYML w.

Syntax Description

w

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

Default 14
Range 5–200

Details

NLDATMYML outputs 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 ++++&quot;;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>max=200 ++++&quot;;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dt nldatmyml.;</td>
<td>November 2012</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.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATMYMMw.

**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 outputs 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.

```
+--- NLDATMYMM min=5 default=11 max=200 ---+
Nov 2012
11/2012
Nov 2012
Nov 2012
+--- NLDATMYMM min=5 default=11 max=200 ---+
```

```
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATMYMM min=5 default=11 max=200 ---+";  
put dt nldatmymm.;  
put dt nldatmymm7.;  
put dt nldatmymm11.;  
put dt nldatmymm200.;  
run;
```

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.

**Category:** Date and Time
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 outputs 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></td>
</tr>
<tr>
<td>dt = datetime();</td>
<td></td>
</tr>
<tr>
<td>dy = date();</td>
<td></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></td>
</tr>
<tr>
<td>put dt nldatmyms5.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dt nldatmyms7.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyms200.;</td>
<td></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.

Category: Date and Time

Alignment: Left

Syntax

NLDATMYQw.
Syntax Description

\( w \)

specifies the width of the output field.

Default: 16

Range: 4–200

Details

The NLDATMYQ format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 67 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><strong>options locale=fr_FR;</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **data _null_;**                    | ![Image of the results](image)
| dy=today();                         | ![Image of the results](image)
| dt=datetime();                      | ![Image of the results](image)
| put "+--- NLDATMYQ min=4 default=16 max=200 +++"; | ![Image of the results](image)
| put '16' +5 dt nldatmyq;            | ![Image of the results](image)
| put '4' +5 dt nldatmyq4.;           | ![Image of the results](image)
| put '14' +5 dt nldatmyq14.;         | ![Image of the results](image)
| put '32' +5 dt nldatmyq32.;         | ![Image of the results](image)
| put '200' +5 dt nldatmyq200.;       | ![Image of the results](image)
| run;                                | ![Image of the results](image) |

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.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATMYQL\( w \).
Syntax Description

`w`

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

Default 18

Range 4–200

Details

NLDATMYQL outputs 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></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.

Category: Date and Time

Alignment: Left

Syntax

`NLDATMYQM w`.

Syntax Description

`w`

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

Default 7
Details

NLDATMYQM outputs 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.

```
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATMYQM min=4 default=7 max=200 ---+";  
put dt nldatmyqm.;  
put dt nldatmyqm6.;  
put dt nldatmyqm7.;  
put dt nldatmyqm200.;  
run;  
```

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

**Category:** Date and Time  
**Alignment:** Left

Syntax

```
NLDATMYQSw.  
```

**Syntax Description**

`w`

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

**Default** 6  
**Range** 4–200

Details

NLDATMYQS outputs 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+++- NLDATMYQS min=4 default=6 max=200 +++-</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>2012/4</td>
</tr>
<tr>
<td>dy = date();</td>
<td>12/4</td>
</tr>
<tr>
<td>put &quot;++++ NLDATMYQS min=4 default=6 max=200 +++-&quot;;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs4.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs6.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs200.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYR Format

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

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATMYR<sub>w</sub>

**Syntax Description**

<sub>w</sub>

specifies the width of the output field.

**Default** 16

**Range** 2–200

**Example**

This example uses the en_US locale option.
options locale=fr_FR;
  data _null_
  dy=today();
  dt=datetime();
  put "+--- NLDATMYR min=2 default=16
      max=200 ---+
      2008 08 2008
      2008 2008
     
  put dt nldatmyr.;
  put dt nldatmyr2.;
  put dt nldatmyr32.;
  put dt nldatmyr200.;
  run;

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.

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

Syntax

NLDATMYW\[w\].

Syntax Description

\[w\] specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>5–200</td>
</tr>
</tbody>
</table>

Example

This example uses the fr_FR locale option.
## Statements

### Options
```sas
options locale=fr_FR;
```

### Data
```sas
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
```

### Run
```sas
run;
```

## Results

### NLDATMZ Format

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

**Category:** Date and Time  
**Alignment:** Left

### Syntax

```
NLDATMZ w.
```

### Syntax Description

- **w** specifies the width of the output field.
  - Default: 40
  - 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; data test; x=datetime(); put x=nldatmz.; run;</td>
<td>x=18 mars 2011 10 h 40 -0400</td>
</tr>
</tbody>
</table>
NLMNIAED Format

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

Category: Numeric
Alignment: Left

Syntax

NLMNIAEDw,d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

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

Default 3
Range 0–28

Example

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

x=put(-1234.56789,nlmniaed32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AED1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLAED Format” on page 190
NLMNIAUD Format

Writes the monetary format of the international expression for Australia.

**Category:** Numeric

**Alignment:** Left

Syntax

NLMNIAUDw.d

Syntax Description

`w`

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

`d`

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

- **Default:** 2
- **Range:** 0–28

Example

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

```plaintext
x=put(-1234.56789,nlmniaud32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AUD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLAUD Format” on page 191
NLMNIBGN Format

Writes the monetary format of the international expression for Bulgaria.

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

### Syntax

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

### 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.

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

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

### See Also

**Format:**

- “NLMNLBGN Format” on page 192
NLMNIBRL Format

Writes the monetary format of the international expression for Brazil.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

`NLMNIBRL_w.d`

**Syntax Description**

- `w` specifies the width of the output field.
  - Default: 12
  - Range: 8–32
- `d` specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

---

**Example**

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

```plaintext
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><code>put x=;</code></td>
<td>(BRL1,234.57)</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

- “NLMNLBRL Format” on page 193
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.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>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 194
NLMNICHF Format

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

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

### Syntax

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

### 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, nlmnichf32.2);
y=put(-1234.56789, 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>(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 195
NLMNICNY Format

Writes the monetary format of the international expression for China.

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

### Syntax

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

### Syntax Description

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

- **d**
  - specifies 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.

\[
x=\text{put}(-1234.56789,\text{nlmnicny32.2});
y=\text{put}(-1234.56789,\text{dollar32.2});
\]

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

### See Also

**Format:**

- “NLMNLCNY Format” on page 196
NLMNICZK Format

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

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

---

**Syntax**

NLMNICZK<em>w.d</em>

**Syntax Description**

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

<em>d</em>  
- 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,nlmniczk32.2);
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:

- “NLMNLCZK Format” on page 197
**NLMNIDKK Format**

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

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

**Syntax**

```
NLMNIDKKw.d
```

**Syntax Description**

- `w` specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32
- `d` specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 2
  - **Range:** 0–28

**Example**

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

```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>(DKK 1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNLDKK Format” on page 198
**NLMNIEEK Format**

Writes the monetary format of the international expression for Estonia.

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

**Syntax**

```
NLMNIEEKw.d
```

**Syntax Description**

- `w` specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32
- `d` specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 4
  - **Range:** 0–28

**Example**

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

```plaintext
x=put(-1234.56789,nlmnieek32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td>$(EEK1,234.57)$</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNIEEK Format” on page 199
NLMNIEGP Format

Writes 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 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 200
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

NLMNIEURw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default 12
Range 8–32

\(d\)

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

Default 2
Range 0–28

Example

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

\(x=\)put(-1234.56789,nlmnieur32.2);
\(y=\)put(-1234.56789,nlmnieur32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>x=-1.234,57 EUR</td>
</tr>
<tr>
<td>put y=;</td>
<td>y=-1.234,57 €</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLEUR Format” on page 201
NLMNIGBP Format

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

Category: Numeric
Alignment: Left

Syntax

NLMNIGBPw.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, nlmnigbp32.2);
y=put(-1234.56789, dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>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 202
**NLMNIHKD Format**

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

 **Category:** Numeric

 **Alignment:** Left

---

**Syntax**

NLMNIHKD\(_w.d\)

**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, nlmnihkd32.2);
y=put(-1234.56789, dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>&quot;$12,345.6789&quot;</td>
</tr>
<tr>
<td>put y=;</td>
<td>&quot;$-1,234.57&quot;</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- "NLMNLHKD Format" on page 203
NLMNIHRK Format

Writes 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 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,nlmnihrk32.2);
y=put(-1234.56789,dollar32.2);

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

See Also

Format:

- “NLMNLHRK Format” on page 204
NLMNIHUF Format

Writes 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 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.

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

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

See Also

Format:
- “NLMNLHUF Format” on page 205
**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.

\[x=\text{put}(-1234.56789,\text{nlmniidr32.2});\]
\[y=\text{put}(-1234.56789,\text{dollar32.2});\]

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

---

**See Also**

**Format:**

- “NLMNLIDR Format” on page 206
**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.

```plaintext
x=put(-1234.56789,nlmniils32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(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 207
NLMNIINR Format

Writes the monetary format of the international expression for India.

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

---

**Syntax**

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

**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.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = );</td>
<td>(INR1,234.57)</td>
</tr>
<tr>
<td>put ( y = );</td>
<td>$-1,234.57 $</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLINR Format” on page 208
**NLMNIJPY Format**

Writes the monetary format of the international expression for Japan.

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

---

**Syntax**

\( \text{NLMNIJPY} \, w.d \)

**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`.

\[ x=\text{put}(-1234.56789,\text{nlmnijpy32.2}); \]
\[ y=\text{put}(-1234.56789,\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>----+----1----+</td>
</tr>
<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 209
**NLMNIKRW Format**

 Writes 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 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.

\[ \begin{align*}
x & = \text{put}(-1234.56789, \text{nlmnikrw32.2}); \\
y & = \text{put}(-1234.56789, \text{dollar32.2});
\end{align*} \]

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

---

**See Also**

**Format:**

- “NLMNLKRW Format” on page 210
NLMNILTL Format

Writes the monetary format of the international expression for Lithuania.

Category: Numeric
Alignment: Left

Syntax

NLMNILTLw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

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

Default 4
Range 0–28

Example

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

x=put(-1234.56789,nlmniltl32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LTL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNLLTL Format” on page 211
**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.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{put } x=;)</td>
<td>((\text{LVL1},234.57))</td>
</tr>
<tr>
<td>(\text{put } y=;)</td>
<td>(-1,234.57)</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLLVL Format” on page 212
**NLMNIMOP Format**

Writes the monetary format of the international expression for Macau.

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

## Syntax

NLMNIMOP\(w.d\)

### 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,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 213
NLMNIMXN Format

Writes the monetary format of the international expression for Mexico.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\( \text{NLMNIMXN}w.d \)

**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`.

\( x=\text{put}(-1234.56789,\text{nlmnimxn}32.2) ; \)

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x=; )</td>
<td>(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 214
**NLMNIMYR Format**

Writes the monetary format of the international expression for Malaysia.

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

---

**Syntax**

\texttt{NLMNIMYRw.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,nlmnimyr32.2);}  
\texttt{y=put(-1234.56789,dollar32.2);}  

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>{MYR1,234.57}</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “\texttt{NLMNLMYR Format}” on page 215
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:**

- “NLMNLNOK Format” on page 216
NLMNINZD Format
Writes the monetary format of the international expression for New Zealand.

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

### Syntax

NLMNINZD<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:** 2
- **Range:** 0–28

### Example

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

\[ x = \text{put}(-1234.56789, \text{nlmninzd32.2}); \]
\[ y = \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NZD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLNZD Format” on page 217
NLMNIPLN Format

Writes 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 the number of digits to the right of the decimal point in the numeric value.
Default 2
Range 0–28

Example

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

x=put(-1234.56789, nlmnipln32.2);
y=put(-1234.56789, dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(PLN1234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLPLN Format” on page 218
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.

\(x=\text{put}(-1234.56789,\text{nlmniirub}32.2)\);

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

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

---

**See Also**

**Format:**

- “NLMNLIRUB Format” on page 219
**NLMNISEK Format**

Writes the monetary format of the international expression for Sweden.

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

---

**Syntax**

\texttt{NLMNISEK}w.d

**Syntax Description**

\textit{w}

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\textit{d}

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

- **Default:** 2
- **Range:** 0–28

---

**Example**

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

\begin{verbatim}
  x=put(-1234.56789,nlmnisek32.2);
  y=put(-1234.56789,dollar32.2);
\end{verbatim}

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

---

**See Also**

**Format:**
- “NLMNLSEK Format” on page 220
NLMNISGD Format

Writes the monetary format of the international expression for Singapore.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\texttt{NLMNISGDw.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,nlmnisgd32.2);
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>(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 221
NLMNITHB Format

Writes the monetary format of the international expression for Thailand.

**Syntax**

\[ \text{NLMNITHB}_{w,d} \]

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

\( d \)

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

- **Default**: 2
- **Range**: 0–28

**Example**

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

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = );</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put ( y = );</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

**Format**:

- “NLMNLTHB Format” on page 222
NLMNITRY Format

Writes 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 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,nlmnitry32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(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 223
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.

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

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

See Also

Format:
- “NLMNLTWD Format” on page 224
**NLMNIUSD Format**

Writes 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:** 912  
**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`.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(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 225
NLMNIZAR Format

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

Category: Numeric
Alignment: Left

Syntax

NLMNIZARw,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,nlnizar32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ZAR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNLZAR Format” on page 226
NLMNLAED Format

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

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\texttt{NLMNLAEDw.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:** 3
- **Range:** 0–28

---

**Example**

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

\begin{verbatim}
x=put(-1234.56789,nlmnlaid32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

---

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=;} & (AED1,234.57) \\
\texttt{put y=;} & $-1,234.57$ \\
\hline
\end{tabular}

---

**See Also**

**Format:**

- “NLMNIAED Format” on page 153
**NLMNLAUD Format**  
Writes the monetary format of the local expression for Australia.

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

---

**Syntax**

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

**Syntax Description**

\[ w \]

specifies the width of the output field.

Default: 12  
Range: 8–32

\[ d \]

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

Default: 2  
Range: 0–28

---

**Example**

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

\[
x = \text{put}(-1234.56789, \text{nlmnlaud32.2});
\]

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

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

---

**See Also**

**Format:**

- “\texttt{NLMNIAUD Format}” on page 154
NLMNLBGN Format

Writes the monetary format of the local expression for Bulgaria.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLBGN \( \text{w}.d \)

**Syntax Description**

\( \text{w} \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( \text{d} \)

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

- **Default:** 2
- **Range:** 0–28

---

**Example**

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

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put x=;}</td>
<td>{BGN1,234.57}</td>
</tr>
<tr>
<td>\text{put y=;}</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

Format:

- “NLMNIBGN Format” on page 155
NLMNLBRL Format

Writes the monetary format of the local expression for Brazil.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLBRL\(w.d\)

**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,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 156
**NLMNLCAD Format**

Writes the monetary format of the local expression for Canada.

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

---

**Syntax**

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

**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`.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(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 157
NLMNLCHF Format

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

**Category:** Numeric

**Alignment:** Left

### Syntax

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

### 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`.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>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 158
NLMNLCNY Format

Writes the monetary format of the local expression for China.

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

### Syntax

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

### 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,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

- “NLMNICNY Format” on page 159
NLMNLCZK Format

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

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLCZKw.d}

\textit{Syntax Description}

\textbf{\textit{w}}

specifies the width of the output field.

Default 12
Range 8–32

\textbf{\textit{d}}

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

Default 4
Range 0–28

\textbf{Example}

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

\begin{verbatim}
x=put(-1234.56789,nlmnlczk32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>(CZK1,234.57)</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

\textbf{See Also}

Format:

- “NLMNICZK Format” on page 160
**NLMNLDKK Format**

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

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

### Syntax

```plaintext
NLMNLDKK w.d
```

### Syntax Description

**w**

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

**d**

- specifies the number of digits to the right of the decimal point in the numeric value.
- Default: 2
- Range: 0–28

### Example

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

```plaintext
x=put(-1234.56789,nlmnlkke32.2);
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:**
  - “NLMNIDKK Format” on page 161
NLMNLEEK Format

Writes the monetary format of the local expression for Estonia.

Category: Numeric
Alignment: Left

Syntax

NLMNLEEKw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

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

Default 4
Range 0–28

Example

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

\$x=put(-1234.56789,nlmnleek32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Krš1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNIEEK Format” on page 162
NLMNLEGP Format

Writes the monetary format of the local expression for Egypt.

Category: Numeric
Alignment: Left

Syntax

NLMNLEGPr.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,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 163
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.

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

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

---

**See Also**

Format:

- “NLMNIEUR Format” on page 164
**NLMNLGBP Format**

Writes 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 the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

---

**Example**

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

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

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

---

**See Also**

**Format:**

- “NLMNIGBP Format” on page 165
NLMNLHKD Format

Writes 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 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,nlmnlhkd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>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 166
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.

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

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

---

**See Also**

**Format:**
- “NLMNIHRK Format” on page 167
NLMNLHUF Format

Writes the monetary format of the local expression for Hungary.

Category: Numeric
Alignment: Left

Syntax

\[ \text{NLMNLHUF} w \cdot d \]

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,nlmnlhuf32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=};</td>
<td>(Ft1,234.57)</td>
</tr>
<tr>
<td>\texttt{put y=};</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIHUF Format” on page 168
NLMNLIDR Format

Writes the monetary format of the local expression for Indonesia.

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

**Syntax**

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.

```plaintext
x=put(-1234.56789,nlmnlidr32.2);
y=put(-1234.56789,dollar32.2);
```

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

**See Also**

Format:

- “NLMNIIDR Format” on page 169
**NLMNLILS Format**

Writes the monetary format of the local expression for Israel.

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

---

**Syntax**

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

**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,nlmnlis32.2);
y=put(-1234.56789,dollar32.2);
```

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

---

**See Also**

**Format:**

- “NLMNIIILS Format” on page 170
NLMNLINR Format

Writes the monetary format of the local expression for India.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\texttt{NLMNLINRw.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,nlmnlinr32.2);
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>[INR1,234.57]</td>
</tr>
<tr>
<td>put y=;</td>
<td>$$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

- “NLMNIINR Format” on page 171
NLMNLJPY Format

Writes the monetary format of the international expression for Japan.

**Category:** Numeric

**Alignment:** Left

---

### Syntax

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.

```plaintext
x=put(-1234.56789,nlmnljpy32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(JPY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

### See Also

**Format:**

- “NLMNIJPY Format” on page 172
NLMNLKRW Format

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

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

### Syntax

NLMNLKRW\(w.d\)

### Syntax Description

\(w\)
- specifies the width of the output field.  
  **Default:** 12  
  **Range:** 8–32

\(d\)
- specifies the number of digits to the right of the decimal point in the numeric value.  
  **Default:** 0  
  **Range:** 0–28

### Example

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

\begin{verbatim}
x=put(-1234.56789,nlmnlkrw32.2);
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>(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 173
**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

- “NLMNILTL Format” on page 174
NLMNLLVL Format

Wants the monetary format of the local expression for Latvia.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

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

**Syntax Description**

\[ \w \]

- Specifies the width of the output field.
  
  **Default:** 12
  
  **Range:** 8–32

\[ \text{d} \]

- Specifies the number of digits to the right of the decimal point in the numeric value.
  
  **Default:** 4
  
  **Range:** 0–28

---

**Example**

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

\[ \text{x=put(-1234.56789,nlmnllvl32.2);} \]

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Ls1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNILVL Format” on page 175
NLMNLMOP Format

Writes 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 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,nlmnlmop32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(P1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- "NLMNIMOP Format" on page 176
NLMNLMXN Format

 Writes the monetary format of the local expression for Mexico.

 Category: Numeric
 Alignment: Left

Syntax

NLMNLMXNw.d

Syntax Description

w
specifies the width of the output field.

  Default: 12
  Range: 8–32

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

  Default: 2
  Range: 0–28

Example

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

x=put(-1234.56789,nlmnlmxn32.2);
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:

- “NLMNIMXN Format” on page 177
NLMNLMYR Format

Writes the monetary format of the local expression for Malaysia.

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

---

### Syntax

NLMNLMYR \( \text{w.d} \)

### Syntax Description

- \( \text{w} \) specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- \( \text{d} \) specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

### Example

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

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x =</td>
<td>(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 178
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`.

\[ \begin{align*}
  x &= \text{put}(-1234.56789, \text{nlmnllok32.2}); \\
  y &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*} \]

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

---

**See Also**

**Format:**

- “NLMNINOK Format” on page 179
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 LOCALE= system option is set to English_UnitedStates.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(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 180
**NLMNLPLN Format**

Writes the monetary format of the local expression for Poland.

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

---

**Syntax**

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

**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,nlmnlpln32.2);
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:**

- “NLMNIPLN Format” on page 181
NLMNLURU Format

Writes the monetary format of the local expression for Russia.

Category: Numeric
Alignment: Left

Syntax

NLMNLURUBw.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,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:

- “NLMNRUB Format” on page 182
**NLMNLSEK Format**

Writes 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 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 183
NLMNLSGD Format

Writes the monetary format of the local expression for Singapore.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLSGD}_{w.d}

\textbf{Syntax Description}

\texttt{w}

specifies the width of the output field.

Default: 12
Range: 8–32

\texttt{d}

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

Default: 2
Range: 0–28

Example

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

\begin{verbatim}
x=put(-1234.56789,nlmnlsgd32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put x=;} & (SG$1,234.57) \\
\texttt{put y=;} & $-1,234.57 \\
\hline
\end{tabular}

See Also

Format:

- “NLMNISGD Format” on page 184
NLMNLTHB Format

Writes the monetary format of the local expression for Thailand.

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

---

**Syntax**

NLMNLTHB<em>w.d</em>

**Syntax Description**

<em>w</em>

specifies the width of the output field.

Default 12  
Range 8–32

<em>d</em>

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

Default 2  
Range 0–28

---

**Example**

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

```plaintext
x=put(-1234.56789,nlmnlthb32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNITHB Format” on page 185
NLMNLTRY Format

Successfully formats the monetary expression for the local currency.

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

### Syntax

NLMNLTRY<sub>W.D</sub>

### Syntax Description

**W**
- Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

**D**
- Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 4
  - Range: 0–28

### Example

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

```plaintext
x=put(-1234.56789,nlmltry32.2);
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>(YTL1,234.57)</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLMNITRY Format” on page 186
NLMNLTWD Format

Writes the monetary format of the local expression for Taiwan.

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

---

**Syntax**

NLMNLTWD<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:** 2
- **Range:** 0–28

---

**Example**

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

```plaintext
x=put(-1234.56789,nlmnltd32.2);  
y=put(-1234.56789,dollar32.2);
```

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

---

**See Also**

**Format:**

- “NLMNITWD Format” on page 187
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 \texttt{LOCALE=} system option is set to \texttt{English UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnlusd32.2);
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>(US$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

- “NLMNIUSD Format” on page 188
NLMNLZAR Format

 Writes 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 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.

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=);</td>
<td>((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 189
NLMNY Format

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

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

### Syntax

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.

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

Format:
- “NLMNYI Format” on page 228

Informats:
- “NLMNY Informat” on page 501
- “NLMNY Informat” on page 502

NLMNYI Format

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

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

Syntax

NLMNYI\(w,d\)

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 NLMNYI\(w,d\) informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMNYI\(w,d\) format writes numeric values by using the international currency code, and locale-dependent
thousands and decimal separators. The position of international currency code is also locale dependent.

*Note:* The NLMNYIw.d format does not convert currency format. Therefore, the value of the formatted number should equal the currency of the current locale value.

**Comparisons**

The NLMNYw.d and NLMNYIw.d formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYIw.d format uses three-letter international currency codes, such as USD, while NLMNYw.d format uses local currency symbols, such as $.

**Example**

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

```plaintext
x=put(-1234.56789,nlmnyi32.2);
y=put(-1234.56789,nlmny32.2);
z=put(-1234.56789,dollar32.2);
```

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

**See Also**

**Format:**

- “NLMNY Format” on page 227

**Informats:**

- “NLMNY Informat” on page 501
- “NLMNYI Informat” on page 502

---

**NLNUM Format**

_Writes the numeric format of the local expression in the specified locale._

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLNUMw.d
**Syntax Description**

`w` specifies the width of the output field.

- **Default**: 6
- **Range**: 1–32

`d` specifies to divide the number by $10^d$. If the data contains decimal separators, the `d` value is ignored.

- **Default**: 0
- **Range**: 0–31

**Details**

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

**Comparisons**

The NLNUMw.d format writes the numeric value with locale-dependent thousand and decimal separators. The NLNUMIw.d format writes the numeric value with a comma (,) as thousands separator and a period (.) as a decimal separator.

If the `w` or `d` values are not large enough to generate a formatted number, the NLNUMw.d format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

**Example**

```plaintext
x=put(-1234356.7891,nlnum32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options LOCALE=English_UnitedStates;</code></td>
<td><code>-1,234,356.79</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
<tr>
<td><code>options LOCALE=German_Germany;</code></td>
<td><code>-1.234,356,79</code></td>
</tr>
<tr>
<td><code>put x=;</code></td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Format:
- “NLNUMI Format” on page 231
Informats:

• “NLNUM Informat” on page 504
• “NLNUMI Informat” on page 505

**NLNUMI Format**

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

**Category:** Numeric

**Alignment:** Left

**Syntax**

`NLNUMIw,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>6</td>
<td>1–32</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>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0–31</td>
</tr>
</tbody>
</table>

**Details**

The NLNUMIw,d informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLNUMIw.d format writes numeric values by using a comma (,) as thousands separator and a period (.) as a decimal separator for all locales.

**Comparisons**

The NLNUMIw,d format writes the numeric data of the international expression in the specified locale. The NLNUMIw,d format writes the numeric value with a comma (,) as thousands separator and a period (.) as a decimal separator.

If the $w$ or $d$ values are not large enough to generate a formatted number, the NLNUMIw,d format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

**Example**

```plaintext
x=put(-1234356.7891,nlnumi32.2);
```
## NLPCT Format

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

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

### Syntax

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

### 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

---

### See Also

- “NLNUM Format” on page 229
- “NLNUM Informat” on page 504
- “NLNUMI Informat” on page 505
Comparisons

The NLPCTw.d format writes percentage data of the local expression in the specified locale. The NLPCTw.d format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTIw.d format writes the percentage value with a comma (,) as thousands separator and a period (.) as a decimal separator.

The NLPCTw.d format is similar to the PERCENTw.d format except the NLPCTw.d format is locale-specific.

Example

\[
x = \text{put}(-12.3456789, \text{nlpct}32.2);
y = \text{put}(-12.3456789, \text{nlpcti}32.2);
z = \text{put}(-12.3456789, \text{percent}32.2);
\]

<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>-1,234.57%</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>{</td>
</tr>
<tr>
<td>put z=;</td>
<td>1234.57%</td>
</tr>
<tr>
<td>----+-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>options LOCALE=German_Germany;</td>
<td>-1.234,57%</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>{</td>
</tr>
<tr>
<td>put z=;</td>
<td>1234.57%</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLPCTI Format” on page 233

Informats:
- “NLPCT Informat” on page 506
- “NLPCTI Informat” on page 508

NLPCTI Format

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

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

NLPCTI\textsubscript{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 6

Range \(4–32\)

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

The NLPCT\textsubscript{w.d} format is similar to the PERCENT\textsubscript{w.d} format except the NLPCT\textsubscript{w.d} format is locale-specific.

Example

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

\begin{verbatim}
  x=put(-12.3456789,nlpcti32.2);
y=put(-12.3456789,percent32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>{1234.57}</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLPCT Format” on page 232

Informats:

- “NLPCT Informat” on page 506
NLPCTN Format

Produces percentages, using a minus sign for negative values.

Category: Numeric
Alignment: Right

Syntax

NLPCTNw.d

Syntax Description

w
specifies the width of the output field.
Default 6
Range 4–32
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.
Range 0–31
Requirement must be less than w

Details

The NLPCTNw.d format multiplies negative values by 100, adds a minus sign to the beginning of the value, and adds a percent sign (%) to the end of the formatted value.

Example

x=-0.02;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x nlpctn6.;</td>
<td>x=−2%</td>
</tr>
<tr>
<td>put x percentn6.;</td>
<td>x=−2%</td>
</tr>
</tbody>
</table>
NLPCTP Format
Writes locale-specific numeric values as percentages.

Category: Numeric
Alignment: Right

Syntax
NLPCTPw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default \( 6 \)

Range \( 4–32 \)

Tip The width of the output field must account for the percent sign (\% ).

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value. This argument is optional. The thousands separator and decimal symbol for the NLPCTP format is locale-specific.

Range \( 0–31 \)

Requirement must be less than \( w \)

Details
The NLPCTPw.d format multiplies values by 100, formats them, and adds a percent sign (\% ) to the end of the formatted value. The NLPCTPw.d format is similar to the The PERCENTw.d format except that the thousands separator and decimal symbol for the NLPCTPw.d format is locale-specific.

Example

\( x = -0.02; \)

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x nlpctp6;</td>
<td>-2%</td>
</tr>
<tr>
<td>put x percent6;</td>
<td>( 2%)</td>
</tr>
</tbody>
</table>
NLPVALUE Format

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

Category: Numeric
Alignment: Left

Syntax

\texttt{NLPVALUE}_w.d

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 6
Range 3–32

\texttt{d}

specifies to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.

Default 4
Range 1–30

Example

This example uses the german_Germany locale option.

Statements:

``` Sas
options locale=german_germany;
data _null_;  
put "+--- nlpvalue min=3 default=6 max=32 ----+",
x=0.1248;
put x= +5 x pvalue. +5 x nlpvalue.;
put x= +5 x 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;
```

Results:

```
+--- nlpvalue min=3 default=6 max=32 ----
x=0.1248  0.1248  0,1248
x=0.1248  0.1  0,1
x=0.1248  0.12  0,12
x=0.1248  0.125  0,125
```
NLSTRMON Format

Writes the month name in the specified locale.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLSTRMON\(w.d\)

**Syntax Description**

\(w\)

specifies the width of the output field

- **Default:** 20
- **Range:** 200-1

\(d\)

specifies the following:

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

- **Default:** 0
- **Range:** 0-3

**Details**

The NLSTRMON\(w.d\) format writes a SAS value, 1–12 as the name-of-the-month in the specified locale. The following examples use the English_UnitedStates locale.

- 1 = the first month (January)
- 2 = the second month (February)
- 3 = the third month (March)
- 4 = the fourth month (April)
- 5 = the fifth month (May)
- 6 = the sixth month (June)
- 7 = the seventh month (July)
- 8 = the eight month (August)

**See Also**

Format:
- “PVALUE Format” in *SAS Viya Formats and Informats: Reference*
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>; monnum = 1; /* January=1, December=12 <em>/ put monnum NLSTRMON20.; put monnum NLSTRMON20.1; /</em> decimal .1 specified use abbreviation. */ put monnum NLSTRMON20.2; put monnum NLSTRMON20.3; run;</td>
<td>January Jan JANUARY JAN</td>
</tr>
</tbody>
</table>

NLSTRQTR Format

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

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

Syntax

`NLSTRQTRw.d`

**Syntax Description**

- **w**
  - specifies the width of the output field
  - Default: 20
  - Range: 1–200

- **d**
  - specifies the following:
    - 00000001: write abbreviated form.
    - 00000010: write capitalized form.
  - Default: 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>1st quarter</td>
</tr>
<tr>
<td>qtrnum = 1 ; //* January=1, December=12 */</td>
<td>Q1</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20. ;</td>
<td>1ST QUARTER</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.1; //* decimal .1 specified use abbreviation. */</td>
<td>Q1</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.2;</td>
<td></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.

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

Syntax

NLSTRWKw.d

**Syntax Description**

w
specifies the width of the output field

- **Default:** 20
- **Range:** 1–200
\( d \)
specifies the following:
- 00000001: write abbreviated form.
- 00000010: write capitalized form.

Default 0
Range 0–3

Details
The NLSTRWKw.d format writes a SAS value, 1–7 as the name-of-the-week in the specified locale. The following examples use the English_UnitedStates locale.
- 1 = First day-of-week (Monday)
- 2 = Second day-of-week (Tuesday)
- 3 = Third day-of-week (Wednesday)
- 4 = Fourth day-of-week (Thursday)
- 5 = Fifth day-of-week (Friday)
- 6 = Sixth day-of-week (Saturday)
- 7 = Seventh day-of-week (Sunday)

Example
This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>;</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 specified use abbreviation. */</td>
<td>SUN</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.

Category: Date and Time
Alignment: Left
Syntax

\text{NLTIMAP}_w. \\

\textit{Syntax Description}

\textit{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 Exception for Date and Time Default Widths on page 67 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Comparisons

The NLTIMAP\textit{w}. format is similar to the \textit{TIMEAMPM}_w. format except that the NLTIMAP\textit{w}. 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; put time nltimap.;</td>
<td>4:24:43 PM</td>
</tr>
<tr>
<td>options locale=German_Germany; put time nltimap14.;</td>
<td>4:24:43 nachm</td>
</tr>
</tbody>
</table>

See Also

Format:
- “\textit{NLTIME Format}” on page 242

\textbf{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.
Syntax

NLTIME\textsubscript{w}.

Syntax Description

\textit{w}

specifies the width of the input field.

Default 20

Range 10–200

Comparisons

The NLTIME\textsubscript{w} format is similar to the TIME\textsubscript{w} format except that the NLTIME\textsubscript{w} 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 nltime.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>16.24</td>
</tr>
<tr>
<td>put time nltime.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLTIMAP Format” on page 241

$UCS2B$ 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` 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>data <em>null</em>;</td>
<td>5927</td>
</tr>
<tr>
<td>x = 'X';</td>
<td></td>
</tr>
<tr>
<td>y=put(x,$ucs2b2.);</td>
<td></td>
</tr>
<tr>
<td>put y $hex.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2L Format” on page 246
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 265
- “$UCS2BE Format” on page 245

Informats:
- “$UCS2B Informat” on page 511
$\text{UCS2BE Format}$

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

$\text{UCS2BE}\ w$.

### Syntax Description

$w$

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

- **Default:** 8  
- **Range:** 1–32000

### Details

The $\text{UCS2BE}\ w$ format writes a character string in the encoding of the current SAS session. It processes character strings that are in big-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

### Comparisons

The $\text{UCS2BE}\ w$ format performs processing that is the opposite of the $\text{UCS2B}\ w$ format.

### Example

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| x = '592700410042'x;  
put x $ucs2be.; | AB |
See Also

Formats:

- “$UCS2B Format” on page 243

Informs:

- “$UCS2B Informat” on page 511
- “$UCS2BE Informat” on page 512

$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.

Category: Character
Alignment: Left

Syntax

$UCS2L w:

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8
Range 2–32767

Details

The $UCS2L w. 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 $UCS2L w. format performs processing that is the opposite of the $UCS2LE w. format.

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></td>
<td>1</td>
</tr>
</tbody>
</table>
Statements | Result
--- | ---
data _null_; | 2759
\(x = \text{'\x91'}\); | 
\(y = \text{put}(x,\$ucs2l2.);\) | 
\(\text{put } y \$\text{hex.};\) | 
\(\text{run;}\) |

### See Also

**Formats:**
- “$UCS2B Format” on page 243
- “$UCS2LE Format” on page 247
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 265

**Informats:**
- “$UCS2B Informat” on page 511
- “$UCS2L Informat” on page 513
- “$UCS2LE Informat” on page 514
- “$UCS2X Informat” on page 515
- “$UTF8X Informat” on page 530

---

### $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

\(\$\text{UCS2LE}_w.\)

**Syntax Description**

\(w\)

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

**Default** 8  
**Range** 1–32000
Details

The $\text{UCS2LE}$ 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 $\text{UCS2LE}$ format performs processing that is the opposite of the $\text{UCS2L}$ format.

Example

This example uses UTF-8 encoding.

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

See Also

Format:

- “$\text{UCS2L}$ Format” on page 246

Informats:

- “$\text{UCS2L Informat}$” on page 513
- “$\text{UCS2LE Informat}$” on page 514

$\text{UCS2X}$ 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

$\text{UCS2Xw}$

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 $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>

See Also

Formats:
- “$UCS2B Format” on page 243
- “$UCS2XE Format” on page 250
- “$UCS2L Format” on page 246
- “$UTF8X Format” on page 265

Informats:
- “$UCS2B Informat” on page 511
- “$UCS2L Informat” on page 513
- “$UCS2X Informat” on page 515
- “$UCS2XE Informat” on page 516
- “$UTF8X Informat” on page 530
$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* 1–32000

---

**Details**

The $UCS2XEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in native-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

**Comparisons**

The $UCS2XEw. format performs processing that is the opposite of the $UCS2Xw. format.

**Example**

This example uses UTF-8 encoding.

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

---

**See Also**

Format:

- “$UCS2X Format” on page 248
Informs:

- "$UCS2X Informat" on page 515
- "$UCS2XE Informat" on page 516

$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

$UCSBw:

Syntax Description

w

Specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 4
Range 4–32767

Details

The $UCSBw. 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 $UCSBw. format performs processing that is the opposite of the $UCSBEnw. 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>
$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

$UCS4BE_w.$

### 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>1–32000</td>
</tr>
</tbody>
</table>

### Details

The $UCS4BE_w.$ 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>$\texttt{x ='00005927000004100000042'}$;</td>
<td>$\texttt{AB}$</td>
</tr>
<tr>
<td>put $\texttt{x _ucsb.}$;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- “$UCS4B Format” on page 251

Informat:
- “$UCS4B Informat” on page 517

$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
- **Alignment:** Left

Syntax

$\texttt{UCS4Lw}$

Syntax Description

$w$

 specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

- **Default:** 4
- **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></td>
</tr>
<tr>
<td>x = '衧';</td>
<td>2759</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
- “$UCS4B Format” on page 251
- “$UCS4LE Format” on page 255
- “$UCS4X Format” on page 256
- “$UTF8X Format” on page 265

Informats:
- “$UCS2B Informat” on page 511
- “$UCS2L Informat” on page 513
- “$UCS2X Informat” on page 515
- “$UCS4B Informat” on page 517
- “$UCS4L Informat” on page 518
- “$UCS4X Informat” on page 519
- “$UTF8X Informat” on page 530
$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 1–32000

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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '2759000041000004200000'x; put x $ucs4le.;</td>
<td>AB</td>
</tr>
</tbody>
</table>

See Also

Format:
- “$UCS4L Format” on page 253
$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.

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

**Details**

The $UCS4X_w.$ 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 $UCS4X_w.$ format performs processing that is the opposite of the $UCS4XE_w.$ format. If you are exchanging data within the same operating environment, use the $UCS4X_w.$ format. If you are exchanging data with a different operating environment, use the $UCS4B_w.$ format or $UCS4L_w.$ format.

**Example**

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '𐍈';</td>
<td>'00005927'x (binary) or '27590000'x (little endian)</td>
</tr>
<tr>
<td>put x $ucs4x4.;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:
- “$UCS2L Format” on page 246
- “$UCS4XE Format” on page 257
- “$UCS2X Format” on page 248
- “$UCS4B Format” on page 251
- “$UCS4L Format” on page 253
- “$UTF8X Format” on page 265

Informats:
- “$UCS2B Informat” on page 511
- “$UCS2L Informat” on page 513
- “$UCS2X Informat” on page 515
- “$UCS4B Informat” on page 517
- “$UCS4B Format” on page 251
- “$UCS4L Informat” on page 518
- “$UCS4X Informat” on page 519
- “$UTF8X Informat” on page 530

$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 1–32000
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 = '27590000410000042000000'x;</td>
<td>AB (little endian)</td>
</tr>
<tr>
<td>put x $ucs4be4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also
Format:
- “$UCS4X Format” on page 256

Informat:
- “$UCS4X Informat” on page 519

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

Category: Character
Alignment: Left

Syntax
$UESCw:

Syntax Description
w specifies the width of the input field.
Default 8
$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

$UESCE_{w}$

---

**Range**

1–32000

---

**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>$x='\u001A' ;$</td>
<td>¥u5927</td>
</tr>
<tr>
<td>$y='\u001B' ;$</td>
<td>¥u5927</td>
</tr>
<tr>
<td>$z='\u001C' ;$</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:**

- “$UESC Informat” on page 522
- “$UESCE Informat” on page 523
**Syntax Description**

\( w \)

specifies the width of the output field.

---

**Default**

8

**Range**

1–32000

---

**Details**

If the data is not supported by the encoding of the current SAS session, the data remains in UESC.

**Comparisons**

The $UESCEw. format performs processing that is the opposite of the $UESCw. format.

**Example**

This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put('ビル27', $uesce10.) ;</td>
<td>x=ビル27</td>
</tr>
<tr>
<td>x=put('ビル27', $uesce10.) ;</td>
<td>x=ビル27</td>
</tr>
<tr>
<td>x=put('ビルビル27', $uesce10.) ;</td>
<td>x=ビルビル27</td>
</tr>
</tbody>
</table>

**See Also**

- “$UESC Format” on page 258

**Informats:**

- “$UESC Informat” on page 522
- “$UESCE Informat” on page 523

---

**$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

$UNCRw

w

specifies the width of the output field.

Default 8

Range 1–32000

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 '¥' 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

Informats:

- “$UNCR Informat” on page 524
- “$UNCRE Informat” on page 525

$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\n
specifies the width of the output field.

Default 8
Range 1–32000

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=&quot;&amp;22823;abc&quot;;</td>
<td>abc</td>
</tr>
<tr>
<td>put x $uncr10.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UNCR Format” on page 260

Informats:
- “$UNCR Informat” on page 524
- “$UNCRE Informat” on page 525

$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 27–32000

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='ä';</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:
• “$UPARENE Format” on page 264

Informats:
• “SUPAREN Informat” on page 526
• “SUPARENE Informat” on page 527
$UPARENE 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**

$UPARENEw$

---

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 8
- **Range:** 1–32000

---

**Comparisons**

The $UPARENEw$ 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 $uparene4.;</td>
<td>abc3</td>
</tr>
</tbody>
</table>

---

**See Also**

**Formats:**
- “$SUPAREN Format” on page 262

**Informats:**
- “$SUPAREN Informat” on page 526  
- “$SUPARENE Informat” on page 527
$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

$UTF8X_{w}.$

### 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.

**Default** 8

**Range** 2–32767

### Comparisons

This example uses UTF-8 encoding.

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

**See Also**

**Formats:**

- “$UCS2B Format” on page 243
- “$UCS2L Format” on page 246
- “$UCS2X Format” on page 248

**Informats:**

- “$UCS2B Informat” on page 511
- “$UCS2L Informat” on page 513
- “$UCS2X Informat” on page 515
$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

$UTF8XEw.

Syntax Description

w specifies the width of the output field. Specify enough width to include all of the characters in the variable. The width of the characters is dependent on the code point value of the individual characters.

Default 8
Range 1–32000

Comparisons

The $UTF8XEw. format performs processing that is the opposite to the $UTF8Xw. 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>

See Also

Formats:
- “$UTF8X Format” on page 265

Informatas:
- “$UTF8X Informat” on page 530
WEEKU Format

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

Category: Date and Time
Alignment: Left

Syntax

\textbf{WEEKU}_{\textit{w}}.

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

Default 11
Range 3–200

Details

The \texttt{WEEKU}_{\textit{w}}. format writes a week-number format. The \texttt{WEEKU}_{\textit{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>yywww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yywwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyywwwdd</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>

Comparisons

The \texttt{WEEKV}_{\textit{w}}. 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 \texttt{WEEKW}_{\textit{w}}. format writes the week number of the year as a decimal number in the range 00–53, with
Monday as the first day of week 1. The WEEKUw. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Example

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,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>
</tbody>
</table>
```

See Also

Formats:

- "WEEKV Format" on page 268
- "WEEKW Format" on page 270

WEEKV Format

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

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

```
WEEKV w:
```

**Syntax Description**

`w` specifies the width of the output field.

**Default:** 11  
**Range:** 3–200
Details

The WEEKVw. format writes the various formats depending on the specified width. Algorithm V calculates the SAS date value, and the number-of-the-week value is represented as a decimal number in the range 01–53, with a leading zero and maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For example, the fifth week of the year would be represented as 06.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yyWwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWwwdd</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>

Comparisons

The WEEKVw. format writes the week number as a decimal number in the range 01–53, with weeks that begin on a Monday and week 1 of the year including both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKWw. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The WEEKUw. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Example

sasdate='01JAN2003'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,weekv3.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate,weekv5.);</td>
<td>03W01</td>
</tr>
<tr>
<td>x=put(sasdate,weekv7.);</td>
<td>03W0103</td>
</tr>
<tr>
<td>y=put(sasdate,weekv9.);</td>
<td>2003W0103</td>
</tr>
<tr>
<td>z=put(sasdate,weekv11.);</td>
<td>2003-W01-03</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>
WEEKW Format

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

- **Category:** Date and Time
- **Alignment:** Left

**Syntax**

WEEKW{w}.

**Syntax Description**

{w} specifies the width of the output field.

- **Default:** 11
- **Range:** 3–200

**Details**

The WEEKW{w} format writes the various formats depending on the specified width. Algorithm W calculates the SAS date value using the number of the week within the year (Monday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yyWwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWwwdd</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>
Comparisons

The WEEKVw. format writes the week number as a decimal number in the range 01–53. 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 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,weekw1.);</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 267
- “WEEKV Format” on page 268

YEN Format

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

- **Category:** Numeric
- **Alignment:** Right

Syntax

YENw.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.

- **Range**: 0–9

**Details**

The 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>--------</td>
<td>1,254.71</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.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

`YYWEEKUw:`
**Syntax Description**

\( w \)

Specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>3-8</td>
</tr>
</tbody>
</table>

**Details**

The YYWEEKU\( w \) format writes a week-number format. The YYWEEKU\( w \) 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).

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-WWWW</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

**Comparisons**

The YYWEEKU\( w \) format is similar to the WEEKU\( w \) format except that the YYWEEKU\( w \) format does not specify the day-of-week information. Also, the YYWEEKU\( w \) format does not accept any 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>----</td>
<td>-------</td>
</tr>
</tbody>
</table>

```
### YYWEEKV Format

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

**Category:** Date and Time  
**Alignment:** Left

#### Syntax

`YYWEEKVw;`

#### Syntax Description

`w`

specifies the width of the output field.

- **Default:** 7  
- **Range:** 3–8

#### 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:
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

\[ \text{sasdate } = \text{ '01JAN2007' } \text{d;} \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u=\text{put(sasdate,yyweekv3.);} )</td>
<td>W01</td>
</tr>
<tr>
<td>( v=\text{put(sasdate,yyweekv4.);} )</td>
<td>W01</td>
</tr>
<tr>
<td>( w=\text{put(sasdate,yyweekv5.);} )</td>
<td>07W01</td>
</tr>
<tr>
<td>( x=\text{put(sasdate,yyweekv6.);} )</td>
<td>07W01</td>
</tr>
<tr>
<td>( y=\text{put(sasdate,yyweekv7.);} )</td>
<td>2007W01</td>
</tr>
<tr>
<td>( z=\text{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 268

YYWEEKW Format

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

Category: Date and Time

Alignment: Left
Syntax

YYWEEK\textsubscript{W}w.

Syntax Description

\textit{w}

specifies the width of the output field.

Default 7

Range 3–8

Details

The YYWEEK\textsubscript{W}w. format writes the various formats depending on the specified width. Algorithm W calculates the SAS date value using the number of the week within the year.

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 YYWEEK\textsubscript{W}w. format is similar to the WEEK\textsubscript{W}w. format except that the YYWEEK\textsubscript{W}w. format does not specify the day-of-week information. Also, the YYWEEK\textsubscript{W}w. format does not accept any width that is greater than 8.

Example

\texttt{sasdate = '01JAN2007'd}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-------</td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
u=put(sasdate,yyweekw3.); | W01
v=put(sasdate,yyweekw4.); | W01
w=put(sasdate,yyweekw5.); | 07W01
x=put(sasdate,yyweekw6.); | 07W01
y=put(sasdate,yyweekw7.); | 2007W01
z=put(sasdate,yyweekw8.); | 2007-01
put u;
put v;
put w;
put x;
put y;
put z;

See Also

Format:
- “WEEKW Format” on page 270
Part 6

Functions for NLS

Chapter 11

Internationalization Compatibility for SAS String Functions .......... 281

Chapter 12

VARCHAR Support in Functions ........................................... 301

Chapter 13

NLS Functions for SAS Only .................................................. 305

Chapter 14

NLS Functions for SAS and the CAS Server ................................. 375
Chapter 11
Internationalization Compatibility for SAS String Functions

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 one 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 multi-byte character set (MBCS), the data is often handled improperly and produce incorrect results.

DBCS encodings require a varying number of bytes to represent each character. MBCS is sometimes used as a synonym for DBCS.

To solve this problem SAS introduced a set of string functions and CALL routines, called K functions, for those string manipulations where DBCS and MBCS data must be handled carefully. This page 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.

The user needs to understand the difference between byte-based offset-length and character-based offset-length in order to use the K functions properly. Most K functions require the character-based offset or length. Under SBCS environments, the byte-based unit is identical to character-based unit. However, under DBCS or MBCS environment, there are significant differences, and programmers need to distinguish them. The users might need to change the programming logic in order to use the K functions. Most K functions require strings encoded in current SAS session encoding. UTF-8 is the only SAS session encoding supported by SAS Viya.

Here is an example of the functionality of a SAS string function and its corresponding K Function.

In a double-byte encoding such as Shift-JIS, this string, STR='8361836262'x has 5 bytes and represents three characters. In a single-byte encoding such as LATIN1, this same string represents five characters as shown in . Table 11.1 on page 282.
Table 11.1  String Representation in Latin1 and Shift-JIS Encoding

<table>
<thead>
<tr>
<th>Shift-JIS Characters</th>
<th>ᐂ</th>
<th>゚</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexadecimal Representation</td>
<td>83</td>
<td>61</td>
<td>83</td>
</tr>
<tr>
<td>Latin1 Characters</td>
<td>ｆ</td>
<td>a</td>
<td>ｆ</td>
</tr>
</tbody>
</table>

The following example uses this same hexadecimal string in the INDEX functions, which returns a value of 4 for both the SBCS and the multi-byte environments. However, the result is incorrect for the multi-byte environment because the sequence of characters 'bb' does not exist in the string in a DBCS encoding. In a multi-byte environment, the result should be 0.

With the SCAN function, the expected returned string would be the first two Japanese characters. However, the function returns only the first one followed by another character that does not even seem to be in the string.

Table 11.2  Code Example Using Regular Functions

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>/* SAS program to submit in a MBCS SAS session */</td>
<td>/* Results */</td>
</tr>
<tr>
<td>data <em>null</em> ;</td>
<td>str=E38382E383</td>
</tr>
<tr>
<td>str= &quot;ヂッ b&quot; ;</td>
<td>i=0</td>
</tr>
<tr>
<td>i=index(str, &quot;bb&quot;) ;</td>
<td>s=dzić</td>
</tr>
<tr>
<td>s=scan(str,1,'b') ;</td>
<td>l=7</td>
</tr>
<tr>
<td>l=length(str) ;</td>
<td></td>
</tr>
<tr>
<td>put str= $hex10. /i= /s= l=; run ;</td>
<td></td>
</tr>
</tbody>
</table>

The code in Table 11.3 on page 283 solves this issue, by substituting the INDEX() function, which works only on single-byte data, with the KINDEX() 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 SUBSTR() functions to use the KLENGTH() or KSUBSTR() functions. Because the two functions read the data differently, you will get different results when running this code in a multi-byte environment. The INDEX function is byte-oriented, so it reads the individual bytes of data. The INDEX function finds the hexadecimal byte sequence '6262'x, which it incorrectly interprets as the sequence 'bb' and returns a value of 4. The KINDEX function, is character-based and interprets the first '62'x value as the second byte of the DBCS character rather than as an ASCII 'b'. Therefore, KINDEX returns 0 instead of 4. In a SBCS environment, the KINDEX function works as the INDEX function.
Table 11.3  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 MBCS SAS session */</td>
<td>str=836183626262</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>i=0</td>
</tr>
<tr>
<td>str= &quot;ﾃﾞｯ&quot; ;</td>
<td>s=ﾃﾞｯ</td>
</tr>
<tr>
<td>i=kindex(str, &quot;bb&quot;);</td>
<td></td>
</tr>
<tr>
<td>s=kscan(str,1,'b' ) ;</td>
<td></td>
</tr>
<tr>
<td>l=klength(str) ;</td>
<td></td>
</tr>
<tr>
<td>put str= $hex10. /i= /s= /l=;</td>
<td></td>
</tr>
<tr>
<td>run ;</td>
<td></td>
</tr>
</tbody>
</table>

The K functions work in an SBCS environment. However, 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 DBCS, MBCS, or SBCS. Here are descriptions of the levels:

I18N Level 0

This function is designed for SBCS data. Do not use this function to process DBCS or MBCS data.

I18N Level 1

This function should be avoided, if possible, if you are processing DBCS or MBCS data. The I18N Level 1 functions might not work correctly with DBCS or MBCS encodings under certain circumstances.

I18N Level 2

This function can be used for SBCS, DBCS, and MBCS (UTF-8) data.

Table 11.4  SAS String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>I18N Level 0</th>
<th>I18N Level 1</th>
<th>I18N Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ANORM420 Function” (p. 308)</td>
<td>Returns a normalized string from an input string encoded in EBCDIC420.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ANYALNUM Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for an alphanumeric character, and returns</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“ANYALPHA Function” in 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></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 digit, and returns the first position at which the digit is found.</td>
<td></td>
<td></td>
<td>X</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></td>
<td>X</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></td>
<td>X</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></td>
<td>X</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></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>“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>“ANYXDIGIT 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. 376)</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 SAS Viya Functions and CALL Routines: Reference</td>
<td>Concatenates character or numeric values by using a delimiter to separate items and by adding quotation marks to strings that contain the delimiter.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“CATS Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Removes leading and trailing blanks, and returns a concatenated character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“CATT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Removes trailing blanks, and returns a concatenated character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“CATX Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Removes leading and trailing blanks, inserts delimiters, and returns a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“CHAR Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a single character from a specified position in a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“CHOOSEC Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a character value that represents the results of choosing from a list of arguments.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“CHOOSEN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a numeric value that represents the results of choosing from a list of arguments.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“COALESCEC Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the first nonmissing value from a list of numeric arguments.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“COLLATE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a character string in 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>“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></td>
<td>X</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></td>
<td>X</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></td>
<td>X</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></td>
<td>X</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></td>
<td>X</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></td>
<td>X</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></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>“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></td>
<td>X</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></td>
<td>X</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. 315)</td>
<td>Returns the current locale/language environment.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLANGUAGE Function” (p. 316)</td>
<td>Returns the current two-letter language code.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLOCALE Function” (p. 317)</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. 318)</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>X</td>
<td></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>X</td>
<td></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 <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Returns a numeric value based on whether an expression is true, false, or missing.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“INDEX Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“INDEXC Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Searches a character expression for any of the specified characters, and returns the position of that character.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“INDEXW Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Searches a character expression for a string that is specified as a word, and returns the position of the first character in the word.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOMPARE Function” (p. 319)</td>
<td>Returns the result of a comparison of character expressions.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOMPRESS Function” (p. 378)</td>
<td>Removes specified characters from a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOUNT Function” (p. 379)</td>
<td>Returns the number of double-byte characters in an expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCVT Function” (p. 320)</td>
<td>Converts data from one type of encoding data to another encoding data.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KINDEX Function” (p. 380)</td>
<td>Searches a character expression for a string of characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KINDEXB Function” (p. 322)</td>
<td>Searches a character expression for a string of characters.</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>“KINDEXCB Function” (p. 323)</td>
<td>Searches a character expression for specified characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEXC Function” (p. 381)</td>
<td>Searches a character expression for specified characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLEFT Function” (p. 382)</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO-SI.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KLENGTH Function” (p. 383)</td>
<td>Returns the length of an argument.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLOWCASE Function” (p. 384)</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPCASE Function” (p. 324)</td>
<td>Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPCHAR Function” (p. 327)</td>
<td>Converts special characters to normal characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPDATA Function” (p. 328)</td>
<td>Removes or converts unprintable characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KREVERSE Function” (p. 385)</td>
<td>Reverses a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KRIGHT Function” (p. 386)</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. 387)</td>
<td>Selects a specified word from a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSTRCAT Function” (p. 388)</td>
<td>Concatenates two or more character expressions.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KSTRIP Function” (p. 389)</td>
<td>Removes leading and trailing blanks from a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KSUBSTR Function” (p. 390)</td>
<td>Extracts a substring from an argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KSUBSTRB Function” (p. 330)</td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KTRIM Function” (p. 392)</td>
<td>Removes trailing DBCS blanks and SO-SI from character expressions.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KTRUNCATE Function” (p. 331)</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>“KUPCASE Function” (p. 393)</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KUPDATE Function” (p. 394)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KUPDATEB Function” (p. 332)</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>“KUPDATES Function” (p. 395)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KVERIFYB Function” (p. 334)</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>“KVERIFY Function” (p. 397)</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>“LEFT Function” 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>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>“LENGTH Function” 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>“LENGTHC Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, including trailing blanks.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LENGTHM Function” in SAS 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>“LENGTHN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, excluding trailing blanks.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“LOWCASE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td></td>
<td>X</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></td>
<td></td>
<td>X</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></td>
<td></td>
<td>X</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></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLDATE Function” (p. 335)</td>
<td>Converts the SAS date value to the date value of the specified locale by using the date format descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLDATM Function” (p. 338)</td>
<td>Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>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>“NLTIME Function” (p. 340)</td>
<td>Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLITERAL Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Converts a character string that you specify to a SAS name literal.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTALNUM Function” in 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></td>
<td>X</td>
</tr>
<tr>
<td>“NOTALPHA Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for a nonalphabetic character, and returns the first position at which the character is found.</td>
<td></td>
<td></td>
<td>X</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></td>
<td></td>
<td>X</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></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>“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></td>
<td></td>
<td>X</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></td>
<td></td>
<td>X</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></td>
<td>X</td>
</tr>
<tr>
<td>“NOTPRINT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Searches a character string for a nonprintable character, and returns the first position at which that character is found.</td>
<td>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></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>“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>X</td>
<td></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>X</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>X</td>
<td></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>X</td>
<td></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>X</td>
<td></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>X</td>
<td></td>
<td></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>X</td>
<td></td>
<td></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>X</td>
<td></td>
<td></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>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>“PUTC Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Enables you to specify a character format at run time.</td>
<td>X</td>
<td></td>
<td></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>X</td>
<td></td>
<td></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>X</td>
<td></td>
<td></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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“REPEAT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a character value that consists of the first argument repeated n+1 times.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“REVERSE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Reverses a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“RIGHT Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Right-aligns a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SASMSG Function” (p. 342)</td>
<td>Specifies a message from a data set. The returned message is based on the current locale and a specified key.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SASMSGL Function” (p. 345)</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>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SCAN Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns the nth word from a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SETLOCALE Function” (p. 350)</td>
<td>Specifies the locale keys for the current SAS locale.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SHA256 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>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>“SORTKEY Function” (p. 347)</td>
<td>Creates a linguistic sort key.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“SOUNDEX Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Encodes a string to facilitate searching.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SPEDIS Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words.</td>
<td></td>
<td></td>
<td>X</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></td>
<td>X</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></td>
<td>X</td>
</tr>
<tr>
<td>“SUBSTR (left 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>“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 zero.</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 367</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></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>“TRIM Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Removes trailing blanks from a character string, and returns one blank if the string is missing.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TRIMN Function” in <em>SAS Viya Functions and CALL Routines: Reference</em></td>
<td>Removes trailing blanks from character expressions, and returns a string with a length of zero if the expression is missing.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONEDSTNAME Function” (p. 362)</td>
<td>Returns a daylight savings time name.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“TZONEDSTOFF Function” (p. 363)</td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONEID Function” (p. 357)</td>
<td>Returns the current time zone ID.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“TZONENAME Function” (p. 359)</td>
<td>Returns the current standard or daylight savings time, time zone name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONEOFF Function” (p. 360)</td>
<td>Returns the user time zone offset.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONES2U Function” (p. 361)</td>
<td>Converts a SAS date time value to a UTC date time value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONESTTNAME Function” (p. 364)</td>
<td>Returns a standard time zone name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONESTTOFF Function” (p. 365)</td>
<td>Returns the time zone offset value for the specified standard time.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONEU2S Function” (p. 366)</td>
<td>Converts a UTC date time value to a SAS date time value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“UNICODE Function” (p. 368)</td>
<td>Converts Unicode characters to the current SAS session encoding.</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>“UNICODEC Function” (p. 369)</td>
<td>Converts characters in the current SAS session encoding to Unicode characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODELEN Function” (p. 371)</td>
<td>Specifies the length of the character unit for the Unicode data.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODEWIDTH Function” (p. 372)</td>
<td>Specifies the length of a display unit for the Unicode data.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UPCASE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“URLDECODE Function” in SAS Viya Functions and CALL Routines: Reference</td>
<td>Returns a string that was decoded using the URL escape syntax.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“URLENCODE Function” in SAS 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>“WHICHC 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>X</td>
<td></td>
</tr>
</tbody>
</table>
SAS Viya introduces a new VARCHAR data type for character data. Variables created using the VARCHAR data type are varying 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 always assumes that the length represents the number of characters. An offset for a VARCHAR variable always represents a character position. The CHAR data type is most suitable for binary data.

Here is an example of the data types using the SUBSTR function.

\[ \text{substr}(X, 10, 2) \] refers to the 10th and 11th character of X when X is a VarChar. A variable that is assigned a VARCHAR data type of length 10 can hold up to 10 characters in any language.

This table shows the differences between fixed-width and VARCHAR. The CHAR data type use 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 one or multiple bytes. An example of a multi-byte UTF-8 character is the Euro (€) character, which has three bytes. Here is a table that shows the representation of the string €123. The € character has three bytes and each number has one byte.

<table>
<thead>
<tr>
<th>Character</th>
<th>€</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexadecimal value</td>
<td>E2</td>
<td>82</td>
<td>AC</td>
<td>C1</td>
</tr>
</tbody>
</table>
Here is an example comparing the fixed-width data type and the VARCHAR data type. Table 12.1 on page 302 shows results from fixed-width variable using the data €123. Table 12.2 on page 303 shows results using VARCHAR.

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. SUBSTR assigns the string in x to xsub starting at the second byte of the € character. The result is invalid UTF-8 data.

3. The example in the third row uses the INDEX function to locate the 1 in the string. The result is xidx=4.

**Table 12.1  Fixed-Width Character Example**

<table>
<thead>
<tr>
<th>Statements</th>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>x=€123 x=E282AC</td>
</tr>
<tr>
<td>x='€123';</td>
<td>xsub=~123 xsub=82AC31323320</td>
</tr>
<tr>
<td>put x=</td>
<td>xidx=4</td>
</tr>
<tr>
<td>x= $hex6.;</td>
<td></td>
</tr>
<tr>
<td>xsub=substr(x,2);</td>
<td></td>
</tr>
<tr>
<td>put xsub=</td>
<td></td>
</tr>
<tr>
<td>xsub= $hex.;</td>
<td></td>
</tr>
<tr>
<td>xidx=index(x,'1');</td>
<td></td>
</tr>
<tr>
<td>put xidx=;</td>
<td>run;</td>
</tr>
</tbody>
</table>

1. The example in the first row assigns the string to a fixed-width character column. The put statement displays the characters then the hexadecimal value for those characters.

2. The example in the second row uses the SUBSTR function. SUBSTR assigns the string in x to xsub starting at the second byte of the € character.

3. The example in the third row uses the INDEX function to locate the 1 in the string. The result is xidx=2.
### Table 12.2  VARCHAR Character Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Statements</th>
</tr>
</thead>
</table>
| data _null_;  
  length v VARCHAR(6);  
v='€123';  
put v=  
v= $hex.;  
vsub=substr(v,2);  
put vsub=  
vsub= $hex.;  
vidx=index(v,'1');  
put vidx=;  
run; | v=€123  
v=E282AC313233  
vsub=123  
vsub=313233202020202020202020202020202020202020202020  
vidx=2 |
Chapter 13
NLS Functions for SAS Only

Functions by Category

Dictionary

ANORM420 Function .......................................................... 308
ENCODCOMPAT Function ..................................................... 313
ENCODISVALID Function ................................................... 314
GETLOCENV Function ....................................................... 315
GETPXLANGUAGE Function ............................................... 316
GETPXLOCALE Function ................................................... 317
GETPXREGION Function ................................................... 318
KCOMPARE Function .......................................................... 319
KCVT Function ............................................................... 320
KINDEXB Function ............................................................ 322
KINDEXCB Function .......................................................... 323
KPROPCASE Function ....................................................... 324
KPROPCHAR Function ....................................................... 327
KPROPDATA Function ....................................................... 328
KINDEXB Function ............................................................ 322
KINDEXCB Function .......................................................... 323
KTRUNCATE Function ....................................................... 331
KUPDATEB Function .......................................................... 332
KVERIFYB Function .......................................................... 334
NLDATE Function ............................................................. 335
NLDATM Function ............................................................. 338
NLTIME Function ............................................................ 340
SASMSG Function ............................................................. 342
SASMSGL Function ............................................................ 345
SORTKEY Function ........................................................... 347
SETLOCALE Function ......................................................... 350
TZONEID Function .............................................................. 357
TZONENAME Function ....................................................... 359
TZONEOFF Function ............................................................ 360
TZONES2U Function ............................................................ 361
TZONEDSTNAME Function ............................................... 362
TZONEDSTOFF Function .................................................... 363
TZONESTTNAME Function ............................................... 364
TZONESTTOFF Function .................................................... 365
TZONEU2S Function ............................................................ 366
TRANTAB Function .............................................................. 367
UNICODE Function ............................................................ 368
UNICODEC Function ........................................................... 369
UNICODELEN Function ....................................................... 371
UNICODEWIDTH Function ................................................... 372
## Functions by Category

The following categories relate to NLS issues:

### Table 13.1  Categories of NLS Functions

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>processes character data</td>
</tr>
<tr>
<td>Currency Conversion</td>
<td>converts one currency to another currency</td>
</tr>
<tr>
<td>DBCS</td>
<td>processes double-byte character set.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>processes data and time data.</td>
</tr>
<tr>
<td>Locale</td>
<td>processes data based on the specified locale.</td>
</tr>
<tr>
<td>Variable Information</td>
<td>processes variable information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>ANORM420 Function (p. 308)</td>
<td>Returns a normalized string from an input string encoded in EBCDIC420.</td>
</tr>
<tr>
<td></td>
<td>KCVT Function (p. 320)</td>
<td>Converts data from one type of encoding data to another type of encoding data.</td>
</tr>
<tr>
<td></td>
<td>TRANTAB Function (p. 367)</td>
<td>Transcodes data by using the specified translation table.</td>
</tr>
<tr>
<td></td>
<td>UNICODE Function (p. 368)</td>
<td>Converts Unicode characters to the current SAS session encoding.</td>
</tr>
<tr>
<td></td>
<td>UNICODEC Function (p. 369)</td>
<td>Converts characters in the current SAS session encoding to Unicode characters.</td>
</tr>
<tr>
<td></td>
<td>UNICODELEN Function (p. 371)</td>
<td>Specifies the length of the character unit for the Unicode data.</td>
</tr>
<tr>
<td></td>
<td>UNICODEWIDTH Function (p. 372)</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. 335)</td>
<td>Converts the SAS date value to the date value of the specified locale by using the date format descriptors.</td>
</tr>
<tr>
<td></td>
<td>NLDATM Function (p. 338)</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>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>NLTIME Function (p. 340)</td>
<td>Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.</td>
</tr>
<tr>
<td></td>
<td>TZONEID Function (p. 357)</td>
<td>Returns the current time zone ID.</td>
</tr>
<tr>
<td></td>
<td>TZONENAME Function (p. 359)</td>
<td>Returns the current standard or daylight savings time, time zone name.</td>
</tr>
<tr>
<td></td>
<td>TZONEOFF Function (p. 360)</td>
<td>Returns the user time zone offset.</td>
</tr>
<tr>
<td></td>
<td>TZONES2U Function (p. 361)</td>
<td>Converts a SAS date time value to a UTC date time value.</td>
</tr>
<tr>
<td></td>
<td>TZONEDSTNAME Function (p. 362)</td>
<td>Returns a daylight savings time name.</td>
</tr>
<tr>
<td></td>
<td>TZONEDSTOFF Function (p. 363)</td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
</tr>
<tr>
<td></td>
<td>TZONESTTNAME Function (p. 364)</td>
<td>Returns a standard time zone name.</td>
</tr>
<tr>
<td></td>
<td>TZONESTTOFF Function (p. 365)</td>
<td>Returns the time zone offset value for the specified standard time.</td>
</tr>
<tr>
<td></td>
<td>TZONEU2S Function (p. 366)</td>
<td>Converts a UTC date time value to a SAS date time value.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOMPARE Function (p. 319)</td>
<td>Returns the result of a comparison of character expressions.</td>
</tr>
<tr>
<td></td>
<td>KINDEXB Function (p. 322)</td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td></td>
<td>KINDEXCB Function (p. 323)</td>
<td>Searches a character expression for specified characters.</td>
</tr>
<tr>
<td></td>
<td>KPROPCHAR Function (p. 327)</td>
<td>Converts special characters to normal characters.</td>
</tr>
<tr>
<td></td>
<td>KPROPDATA Function (p. 328)</td>
<td>Removes or converts unprintable characters.</td>
</tr>
<tr>
<td></td>
<td>KSUBSTRB Function (p. 330)</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>KTRUNCATE Function (p. 331)</td>
<td>Truncates a string to a specified length in byte unit without breaking multibyte characters.</td>
</tr>
<tr>
<td></td>
<td>KUPDATEB Function (p. 332)</td>
<td>Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.</td>
</tr>
<tr>
<td></td>
<td>KVERIFYB Function (p. 334)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
</tr>
</tbody>
</table>
### Category: Language Elements

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>ENCODCOMPAT Function (p. 313)</td>
<td>Verifies the transcoding compatibility between two encodings.</td>
</tr>
<tr>
<td></td>
<td>ENCODISVALID Function (p. 314)</td>
<td>Verifies a valid encoding name.</td>
</tr>
<tr>
<td>Locale</td>
<td>GETLOCENV Function (p. 315)</td>
<td>Returns the current locale/language environment.</td>
</tr>
<tr>
<td></td>
<td>GETPXLANGUAGE Function (p. 316)</td>
<td>Returns the current two-letter language code.</td>
</tr>
<tr>
<td></td>
<td>GETPXLOCALE Function (p. 317)</td>
<td>Returns the POSIX locale value for a SAS locale.</td>
</tr>
<tr>
<td></td>
<td>GETPXREGION Function (p. 318)</td>
<td>Returns the current two-letter region code.</td>
</tr>
<tr>
<td></td>
<td>SASMSG Function (p. 342)</td>
<td>Specifies a message from a data set. The returned message is based on the current locale and a specified key.</td>
</tr>
<tr>
<td></td>
<td>SASMSGL Function (p. 345)</td>
<td>Specifies a message from a data set. The message is based on a specified locale value and a specified key value.</td>
</tr>
<tr>
<td></td>
<td>SORTKEY Function (p. 347)</td>
<td>Creates a linguistic sort key.</td>
</tr>
<tr>
<td></td>
<td>SETLOCALE Function (p. 350)</td>
<td>Specifies the locale keys for the current SAS locale.</td>
</tr>
<tr>
<td>Variable Information</td>
<td>VARTRANSCODE Function (p. 372)</td>
<td>Returns the transcode attribute of a SAS data set variable.</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 281.
Syntax

ANORM420 (string,<modifiers>)

Required Argument

string
specifies an input string that is encoded in EBCDIC420.

Optional Argument

modifiers
are character constants, variables, or expressions that specify one or more modifiers. The following modifiers can be in uppercase or lowercase:

- g ignores the ligature
- i ignores the Arabic-Indic numbers
- p ignores mapping of shaped characters
- s ignores the addition of space
- t ignores transcoding
- z ignores mapping of the zero-length-space character

Details

The ANORM420 function processes data that is encoded in EBCDIC420. Refer to the following tables for information about the modifier variables:

The ANORM420 function replaces the following character and code points with the appropriate ligature unless the modifier g is specified.

Table 13.2 Modifier g: Arabic Ligatures

<table>
<thead>
<tr>
<th>Character</th>
<th>Unicode Value</th>
<th>From</th>
<th>To</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ﯲ</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>ﯳ</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>ﯴ</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>ﯵ</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>ﯶ</td>
<td>(U+FEFC)</td>
<td>0xB8</td>
<td>0xB1 + 0x56</td>
<td>Arabic Ligature Lam with Aleph - Isolated form</td>
</tr>
<tr>
<td>ﯷ</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 13.3  Modifier \( i \): Arabic-Indic Numbers**

<table>
<thead>
<tr>
<th>From</th>
<th>( \U+0660 )</th>
<th>( \U+0661 )</th>
<th>( \U+0662 )</th>
<th>( \U+0663 )</th>
<th>( \U+0664 )</th>
<th>( \U+0665 )</th>
<th>( \U+0666 )</th>
<th>( \U+0667 )</th>
<th>( \U+0668 )</th>
<th>( \U+0669 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0xDF</td>
<td>0xEA</td>
<td>0xEB</td>
<td>0xED</td>
<td>0xEE</td>
<td>0xFB</td>
<td>0xFC</td>
<td>0xFD</td>
<td>0xFE</td>
<td></td>
</tr>
<tr>
<td>To</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>0xF0</td>
<td>0xF1</td>
<td>0xF2</td>
<td>0xF3</td>
<td>0xF4</td>
<td>0xF5</td>
<td>0xF6</td>
<td>0xF7</td>
<td>0xF8</td>
<td>0xF9</td>
</tr>
</tbody>
</table>

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

**Table 13.4  Modifier \( p \): Shaped Characters Mapping**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x43</td>
<td>U+FE7D</td>
</tr>
<tr>
<td>0x48</td>
<td>FE82</td>
</tr>
<tr>
<td>0x51</td>
<td>U+FE84</td>
</tr>
<tr>
<td>0x57</td>
<td>U+FE8E</td>
</tr>
<tr>
<td>0x59</td>
<td>U+FE91</td>
</tr>
<tr>
<td>0x64</td>
<td>U+FE97</td>
</tr>
<tr>
<td>0x66</td>
<td>U+FE9B</td>
</tr>
<tr>
<td>0x68</td>
<td>U+FE9F</td>
</tr>
<tr>
<td>0x70</td>
<td>U+FEA3</td>
</tr>
<tr>
<td>0x72</td>
<td>U+FEA7</td>
</tr>
<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>
</tbody>
</table>
The ANORM420 function adds a space after the following characters unless the modifier is specified.

**Table 13.5** Modifier: ignores the addition of a space

| From  | To       |阿拉伯字母
|-------|----------|----------------|
| 0x9C  | U+FECB  | 0x9A U+0639
| 0x9D  | U+FECC  | 0x9A U+0639
| 0x9F  | U+FECE  | 0x9E U+063A
| 0xA0  | U+FECF  | 0x9E U+063A
| 0xAA  | U+FED0  | 0x9E U+063A
| 0xAC  | U+FED3  | 0xAB U+0641
| 0xAE  | U+FED7  | 0xAD U+0642
| 0xB0  | U+FEDB  | 0xAF U+0643
| 0xBA  | U+FEDF  | 0xB1 U+0644
| 0xBC  | U+FEE3  | 0xBB U+0645
| 0xBE  | U+FEE7  | 0xBD U+0646
| 0xCB  | U+FEEB  | 0xBF U+0647
| 0xCD  | U+FEEC  | 0xBF U+0647
| 0xDB  | U+FEF0  | 0xDA U+061F
| 0xDD  | U+FEF2  | 0xDC U+064A
| 0xDE  | U+FEF3  | 0xDC U+064A

| From  | To  |阿拉伯字母
|-------|-----|----------------|
| 0x58  | (U+0628) | BEH
| 0x62  | (U+0629) | TEH MARBUTA
| 0x63  | (U+062A) | TEH
| 0x65  | (U+062B) | THEH
| 0x67  | (U+062C) | JEEM
| 0x69  | (U+062D) | HAH
<table>
<thead>
<tr>
<th>Character</th>
<th>Code Point</th>
<th>Unicode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>خ</td>
<td>0x71</td>
<td>(U+062E)</td>
<td>Arabic Letter KHAH</td>
</tr>
<tr>
<td>س</td>
<td>0x77</td>
<td>(U+0633)</td>
<td>Arabic Letter SEEN</td>
</tr>
<tr>
<td>ش</td>
<td>0x80</td>
<td>(U+0634)</td>
<td>Arabic Letter SHEEN</td>
</tr>
<tr>
<td>ص</td>
<td>0x8B</td>
<td>(U+0635)</td>
<td>Arabic Letter SAD</td>
</tr>
<tr>
<td>ض</td>
<td>0x8D</td>
<td>(U+0636)</td>
<td>Arabic Letter DAD</td>
</tr>
<tr>
<td>ع</td>
<td>0x9A</td>
<td>(U+0639)</td>
<td>Arabic Letter AIN</td>
</tr>
<tr>
<td>غ</td>
<td>0x9B</td>
<td>(U+FECA)</td>
<td>Arabic Letter AIN final form</td>
</tr>
<tr>
<td>ع</td>
<td>0x9E</td>
<td>(U+063A)</td>
<td>Arabic Letter GHAIN</td>
</tr>
<tr>
<td>غ</td>
<td>0x9F</td>
<td>(U+FECE)</td>
<td>Arabic Letter GHAIN final form</td>
</tr>
<tr>
<td>ف</td>
<td>0xAB</td>
<td>(U+0641)</td>
<td>Arabic Letter FEH</td>
</tr>
<tr>
<td>ق</td>
<td>0xAD</td>
<td>(U+0642)</td>
<td>Arabic Letter QAF</td>
</tr>
<tr>
<td>ك</td>
<td>0xAF</td>
<td>(U+0643)</td>
<td>Arabic Letter KAF</td>
</tr>
<tr>
<td>ل</td>
<td>0xB1</td>
<td>(U+0644)</td>
<td>Arabic Letter LAM</td>
</tr>
<tr>
<td>م</td>
<td>0xBB</td>
<td>(U+0645)</td>
<td>Arabic Letter MEEM</td>
</tr>
<tr>
<td>ن</td>
<td>0xBD</td>
<td>(U+0646)</td>
<td>Arabic Letter NOON</td>
</tr>
<tr>
<td>ه</td>
<td>0xBF</td>
<td>(U+0647)</td>
<td>Arabic Letter HEH</td>
</tr>
<tr>
<td>ي</td>
<td>0xDA</td>
<td>(U+0649)</td>
<td>Arabic Letter ALEF MAKSURA</td>
</tr>
<tr>
<td>ي</td>
<td>0xDC</td>
<td>(U+064A)</td>
<td>Arabic Letter YEH</td>
</tr>
<tr>
<td>ي</td>
<td>0xDB</td>
<td>(U+FEF0)</td>
<td>Arabic Letter ALEF MAKSURA final form</td>
</tr>
<tr>
<td>ي</td>
<td>0xDD</td>
<td>(U+FEF2)</td>
<td>Arabic Letter YEH final form</td>
</tr>
</tbody>
</table>

The ANORM420 function transcodes the normalized string to the session encoding unless the modifier \( t \) is specified.
The ANORM420 function remaps the zero-length-break-space character (U+200B), also abbreviated ZWSP, to a space character unless the modifier z is specified. The zero-length character is located at code position 0x45.

**Example**

The following example demonstrates the ANORM420 function:

```
Statements                      Results
data _null_;                   s1=C8E5C7E3C7D320A02020
a = '59CD57BC577745'x ;         s2=C8E5C7E3C7D3A0202020
s1 = anorm420(a) ;              s3=59CD57BC577740454040
/* Turn off addition of space and mapping of   
Arabic-Indic numbers */        
s2 = anorm420(a,"si") ;        
/* Turn off transcoding */      
s3 = anorm420(a,"t") ;        
put s1= $hex20.; / s2= $hex20.; / s3=$hex20.;
run;
```

**ENCODCOMPAT Function**

Verifies the transcoding compatibility between two encodings.

**Category:** Encoding

**Syntax**

ENCODCOMPAT(source1,<source2>)

**Required Arguments**

*source1*

a character string that represents an encoding.

*source2*

a character string that represents an encoding. This argument is optional.

**Details**

If you specify one encoding, the function verifies the compatibility of the specified encoding with the current SAS session encoding.

If you specify two encodings, the function verifies the compatibility of the two encodings.

The function compares two encoding identifiers and determines whether the data needs to be transcoded. *Source1* is the source encoding. *Source2* is the destination encoding. Transcoding 7-bit ASCII to another type of ASCII is compatible, but transcoding ASCII to 7-bit ASCII might not be compatible.

The ENCODCOMPAT function specifies the following values:
−1 Source1 is not a valid encoding name.
−2 Source2 is not a valid encoding name.
0 The encodings are not compatible. Transcoding is needed.
1 The encodings are compatible. Transcoding is not needed.
2 A newline character is detected.

Example

The following examples demonstrate the ENCODCOMPAT features:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>/* session encoding is wlatin1 */</td>
<td>-1</td>
</tr>
<tr>
<td>isCompat= EncodCompat(&quot;xyz&quot;);</td>
<td></td>
</tr>
<tr>
<td>put isCompat;</td>
<td></td>
</tr>
<tr>
<td>/* session encoding is wlatin1 */</td>
<td></td>
</tr>
<tr>
<td>isCompat= EncodCompat (&quot;cp1252&quot;);</td>
<td>1</td>
</tr>
<tr>
<td>put isCompat;</td>
<td></td>
</tr>
<tr>
<td>isCompat= EncodCompat (&quot;ebcdic1149&quot;,&quot;open_ed-1149&quot;);</td>
<td>2</td>
</tr>
<tr>
<td>put isCompat;</td>
<td></td>
</tr>
<tr>
<td>isCompat= EncodCompat (&quot;cp1251&quot;,&quot;ebcdic1149&quot;);</td>
<td>0</td>
</tr>
<tr>
<td>put isCompat;</td>
<td></td>
</tr>
</tbody>
</table>

ENCODISVALID Function

Verifies a valid encoding name.

**Category:** Encoding

**Syntax**

ENCODISVALID(source)

**Required Argument**

source

a character string that represents an encoding name.

**Details**

The ENCODISVALID function returns the following values:

0  the character string is not a valid encoding name.
1  the character string is a valid short encoding name.
the character string is a valid encoding name.
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 281.

**Syntax**
GETLOCENV()  

**Details**
The GETLOCENV function returns the locale/language environment value for a valid SAS locale. The following environment values are possible:

**SBCS**
The SAS session encoding is SBCS (Single-Byte Character Set). SASWZSD is loaded for string manipulation.

**DBCS**
The SAS session encoding is DBCS (Double-Byte Character Set). SASWZSD is loaded for string manipulation.

**MBCS**
The SAS session encoding is Unicode(UTF8). SASWZSU is loaded for string manipulation.
If you receive a blank value, then the WZSS subsystem is not available. This action suggests a configuration or installation error.

**Example**

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>SBCS</td>
</tr>
<tr>
<td>environ=getlocenv();</td>
<td></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 281](#).

**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>fr</td>
</tr>
<tr>
<td>lang=getpxLanguage();</td>
<td></td>
</tr>
<tr>
<td>put lang;</td>
<td></td>
</tr>
<tr>
<td>option locale=German;</td>
<td>de</td>
</tr>
<tr>
<td>lang=getpxLanguage();</td>
<td></td>
</tr>
<tr>
<td>put lang;</td>
<td></td>
</tr>
</tbody>
</table>
option locale=en_US;
lang=getpxLanguage();
put lang;

See Also

System Options:
• “LOCALE System Option” on page 550

Functions:
• “GETPXREGION Function” on page 318
• “GETPXLOCALE Function” on page 317

GETPXLOCALE Function

Returns the POSIX locale value for a SAS locale.

Syntax

GETPXLOCALE(<source>)

Required Argument

<source>

is an optional argument that specifies a locale name.

Details

The GETPXLOCALE function returns the POSIX locale value for a valid SAS locale name. If you specify an invalid locale name, then a null string is returned. If you do not specify a value for the <source> argument, then the function returns the POSIX name for the current SAS session. The length of the POSIX locale name is five characters. If the size of the variable that receives the value is less than five characters, the value is truncated.

Example

In the first example, the LOCALE= system option is set to French_France. In the second example, the <source> argument is set to German_Germany. In the third example, the <source> argument is set to English_United States.
See Also

System Options:

• “LOCALE System Option” on page 550

Functions:

• “GETPXLANGUAGE Function” on page 316
• “GETPXREGION Function” on page 318

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 281.

Syntax

GETPXREGION()
Statements | Results
--- | ---
option locale=french_france;
region=getpxRegion();
put region; | FR

option locale=German;
region=getpxRegion();
put region; | DE

option locale=en_US;
region=getpxRegion();
put region; | US

See Also

System Options:
- “LOCALE System Option” on page 550

Functions:
- “GETPXLOCALE Function” on page 317
- “GETPXLANGUAGE Function” on page 316

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 281.

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

**Syntax**

`KCOMPARE(source,<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 pageno=1 nodate ls=80 ps=60;</td>
<td></td>
</tr>
<tr>
<td>data test;</td>
<td></td>
</tr>
<tr>
<td>rc1 = kcompare('漢字のテスト','漢字');</td>
<td></td>
</tr>
<tr>
<td>rc2 = kcompare('漢字のテスト','テスト');</td>
<td>Obs rc1 rc2 rc3 rc4 rc5</td>
</tr>
<tr>
<td>rc3 = kcompare('漢字のテスト','テスト');</td>
<td>1 3 0 3 1 4</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>

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 281.
Syntax

KCVT(text, intype, outtype, <options, ...>)

Required Arguments

text
specifies the character variable to be converted.

intype
specifies the encoding of the data. The encoding of the text must match the input data's encoding. For valid values, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 613.

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 613.

ASCIIANY and EBCIDICANY are invalid encoding values.

options
specifies character data options. Here are the available options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOSOSI</td>
<td>No shift code or Hankaku characters.</td>
</tr>
<tr>
<td>NOSHIFT</td>
<td></td>
</tr>
<tr>
<td>INPLACE</td>
<td>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.</td>
</tr>
<tr>
<td>KANA</td>
<td>Includes Hankaku katakana characters in columns of character data.</td>
</tr>
<tr>
<td>UPCASE</td>
<td>Converts a 2-byte alphabet to uppercase characters.</td>
</tr>
<tr>
<td>LOWCASE</td>
<td>Converts a 2-byte alphabet to lowercase characters.</td>
</tr>
<tr>
<td>KATA2HIRA</td>
<td>Converts katakana data to hiragana.</td>
</tr>
<tr>
<td>HIRA2KATA</td>
<td>Converts hiragana data to katakana.</td>
</tr>
</tbody>
</table>

Details

See “Internationalization Compatibility for SAS String Functions” on page 281 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.

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

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 281.

**Syntax**

```sas
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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data _null;</td>
<td></td>
</tr>
<tr>
<td>text='漢字文字列の検索';</td>
<td>Result=19</td>
</tr>
<tr>
<td>result=kindexb(text, '検索');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>text2='探';</td>
<td>Result=0</td>
</tr>
<tr>
<td>result=kindexb(text, text2);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

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 281.

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>Result=19</td>
</tr>
<tr>
<td>a='ＡＢＣデフェリ23456';</td>
<td></td>
</tr>
<tr>
<td>result=kindexcb(a, '1234567890','感嘆');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**KPROPCASE Function**

Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.

| Category: | DBCS |
| Restriction: | This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 281. |

**Syntax**

```
str=KPROPCASE(<instr> , (<options> ))
```

**Required Arguments**

str
data string that has been converted and is in the current SAS session encoding.

instr
input data string.

options
converts Japanese, Chinese, Korean, and Taiwanese characters based on specified options.

**HALF-KATAKANA, FULL-KATAKANA**

This option converts half-width katakana to full-width katakana and is used only with Japanese encoding.

Restriction
This option cannot be used at the same time with the full-Katakana, half-Katakana option.

**FULL-KATAKANA, HALF-KATAKANA**

This option converts full-width katakana to half-width katakana and is used only with Japanese encoding.
Restriction  This option cannot be used at the same time with the half-Katakana, full-Katakana option.

**KATAKANA, ROMAJI**
This option converts the katakana character string to a romaji character string and is used only with Japanese encoding.

Restriction  This option cannot be used at the same time with the romaji, katakana option.

**ROMAJI, KATAKANA**
This option converts the romaji character string to a katakana character string and is used only with Japanese encoding.

Restriction  This option cannot be used at the same time with the katakana, romaji option.

**FULL-ALPHABET, HALF-ALPHABET**
This option converts the Full-Alphabet characters to Half-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.

Restriction  This option cannot be used at the same time with the Half-Alphabet, Full-Alphabet option.

**HALF-ALPHABET, FULL-ALPHABET**
This option converts the Half-Alphabet characters to Full-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.

Restriction  This option cannot be used at the same time with the Full-Alphabet, Half-Alphabet option.

**LOWERCASE, UPPERCASE**
This option converts lowercase alphabet characters to uppercase alphabet characters.

Restriction  This option cannot be used at the same time with the Uppercase, Lowercase option.

**UPPERCASE, LOWERCASE**
This option converts uppercase alphabet characters to lowercase alphabet characters.

Restriction  This option cannot be used at the same time with the Lowercase, Uppercase option.

**PROPER**
This option specifies the following default options based on the encoding:

- Japanese encoding
- Half-Katakana, Full-Katakana
- Full-alphabet, Half-alphabet
- Lowercase, Uppercase
- Korean encoding:
- Full-alphabet, Half-alphabet
This function converts the input string based on the specified options and default options. The KPROPCASE function supports the Chinese, Japanese, Korean, Taiwanese (CJKT) environment.

Example

The following example demonstrates the functionality of the KPROPCASE function:

```sas
length fullkana halfkana upper lower fullalpha $ 200;
length str1 str2 str3 str4 str5 str7 str8 $ 30 str6 $44;
lower = 'do-naxtutsu'; /* Doughnuts in Japanese Roman word. */
upper = 'DO-NAXTUTSU'; /* Doughnuts in Japanese Roman word. */
fullkana = unicode('\u30C9\u30FC\u30CA\u30C3\u30C4');
halfkana = unicode('\uFF84\uFF9E\uFF70\uFF85\uFF6F\uFF82');
fullalpha = unicode('\uFF24\uFF2F\uFF0D\uFF2E\uFF21\uFF38\uFF34\uFF33\uFF35\uFF33\uFF35');
str1 = kpropcase(fullkana, 'full-katakana,half-katakana');
if (halfkana EQ trim(str1)) then
   put str1= $hex14.;
str2 = kpropcase(halfkana, 'half-katakana, full-katakana');
if (fullkana EQ trim(str2)) then
   put str2= $hex22.;
str3 = kpropcase(fullkana, 'katakana,romaji');
if (trim(str3) EQ upper) then
   put str3= ;
str4 = kpropcase(upper, 'romaji,katakana');
if (trim(str4) EQ fullkana) then
   put str4= $hex22.;
str5 = kpropcase(fullalpha, 'full-alphabet, half-alphabet');
if (trim(upper) EQ str5) then
   put str5=;
str6 = kpropcase(upper, 'half-alphabet, full-alphabet');
if (trim(str6) EQ fullalpha) then
   put str6= $hex46.;
str7 = kpropcase(lower, 'lowercase, uppercase');
if (trim(str7) EQ upper) then
   put str7=;
str8 = kpropcase(upper, 'uppercase, lowercase');
if (trim(str8) EQ lower) then
   put str8=;
RESULTS:
str1=C4DEB0C5AFC220
str2=8368815B83698362836320
str3=DO-NAXTUTSU
str4=8368815B83698362836320
```
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 281.

**Syntax**

```
str = KPROPCHAR(<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:

```sql
length in1 out1 $30;
  in1=unicode('\u2460\u2473\u277F\u325F');
  out1=KPROPCHAR(in1);
  put out1;
RESULTS:
(1)(20)(-10)(35)
```
KPROPDATA Function

Removes or converts unprintable characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see [Internationalization Compatibility on page 281](#).

**Syntax**

\[
str = \text{KPROPDATA}(<\text{instr}> (<\text{options}>, <\text{input encode name}>, <\text{output encode name}>))
\]

**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 613.

**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 613.

**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:

```sas
length instr $12;
length str1 str2 str3 str4 str5 str6 str7 str8 str9 str10$ 50;
instr = "534153"x||"ae"x || " System";
put instr;
str1 = kpropdata(instr);
put str1= +2 str1= $hex26.;
str2 = kpropdata(instr,'UESC');
put str2= +2 str2= $hex26.;;
str3 = kpropdata(instr, 'UESC','wlatin1');
put str3= +2 str3= $hex34.;
str4 = kpropdata(instr,'TRIM','wlatin1');
put str4= +2 str4= $hex26.;
str5 = kpropdata(instr,'BLANK', 'wlatin1');
put str5= +2 str5= $hex26.;
str6 = kpropdata(instr,'?', 'wlatin1');
put str6= +2 str6= $hex26.;
str7 = kpropdata(instr,'hex', 'wlatin1');
put str7= +2 str7= $hex26.;
str8 = kpropdata(instr,'TRUNC', 'wlatin1');
```
put str8= +2 str8= $hex26.;
str9 = kpropdata(instr,'REMOVE', 'wlatin1');
put str9= +2 str9= $hex26.;
str10 = kpropdata(instr,'NCR', 'wlatin1');
put str10= +2 str10= $hex26.;

RESULTS:
SAS? System
str1=SAS? System   str1=5341533AB2053797374656D2020
str2=SAS? System   str2=5341533AB2053797374656D2020
str3=SAS\uff6e System    str3=5341535C75666D36652053797374656D2020
str4=SAS System       str4=5341532053797374656D202020
str5=SAS System       str5=534153202053797374656D2020
str6=SAS? System      str6=5341533F2053797374656D2020
str7=SAS\xAE System    str7=5341535C784145205379737465
str8=SAS System       str8=5341532020202020202020
str9=               str9=20202020202020202020202020202020202020
str10=SAS® System    str10=53415326233137343B20537973

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 281.

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 281 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>text='漢字文字列の抽出';</td>
<td>result=漢字文</td>
</tr>
<tr>
<td>result=ksubstrb(text,1,9);</td>
<td>result=字文</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>result=ksubstrb(text,3,7);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Function:

- “KSUBSTR Function” on page 390

---

**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 281.

**Syntax**

\[
\text{KTRUNCATE}(\text{argument, number, length})
\]

**Required Arguments**

- **argument** specifies any SAS character expression.
- **number** is numeric.
length

is an integer.

Details

See “Internationalization Compatibility for SAS String Functions” on page 281 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.

<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=ktruncate(&quot;漢字のテスト&quot;,6);</td>
<td>x1=漢字</td>
</tr>
<tr>
<td>x2=ktruncate(&quot;漢字のテスト&quot;,7);</td>
<td>x2=漢字</td>
</tr>
<tr>
<td>x3=ktruncate(&quot;漢字のテスト&quot;,9);</td>
<td>x3=漢字の</td>
</tr>
<tr>
<td>y1=ktruncate(&quot;漢字のテスト&quot;,5,9);</td>
<td>y1=字のテ</td>
</tr>
<tr>
<td>y2=ktruncate(&quot;漢字のテスト&quot;,6,6);</td>
<td>y2=のテ</td>
</tr>
<tr>
<td>y3=ktruncate(&quot;漢字のテスト&quot;,7,6);</td>
<td>y3=のテス</td>
</tr>
<tr>
<td>put x1= / x2= / x3= / y1= / y2= / y3=;</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 281.

Syntax

KUPDATEB(argument, position, n<,characters-to-replace>)

KUPDATEB(argument, position <, n>, characters-to-replace)
**Required Arguments**

*argument*
- specifies a character variable.

*position*
- specifies the beginning character position in byte units.

*n*
- specifies the length of the substring to be replaced in byte units.

**Restrictions**

*n* cannot be larger than the length (in bytes) of the expression that remains in *argument* after *position*.

*n* is optional, but you cannot omit both *n* and *characters-to-replace* from the function.

**Tip**
- If you omit *n*, SAS uses all of the characters in *characters-to-replace* to replace the values of *argument*.

*characters-to-replace*
- specifies a character expression to replace the contents of *argument*.

**Restriction**

*characters-to-replace* is optional, but you cannot omit both *characters-to-replace* and *n* from the function.

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

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 281 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,’がんじ’);</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 394

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 281.

**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, wlati1 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.
**NLDATE Function**

Converts the SAS date value to the date value of the specified locale by using the date format descriptors.

**Category:** Date and Time

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 281.

**Syntax**

\[
\text{NLDATE}(\text{date,descriptor})
\]

**Required Arguments**

- **date**
  - specifies a SAS date 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.

---

**Statements** | **Results**
---|---
\begin{align*}
data \_null\_;  
\text{check}=’漢字’;  
\text{text}=’漢字の検索’;  
x1=\_v\text{erify}(\text{text,check});  
\text{put } x1=;  
x2=\_v\text{erifyh}(\text{text,check});  
\text{put } x2=;  
\text{run;}
\end{align*}

\[x1=3\]
\[x2=7\]
%B  specifies the long-month descriptor. The range for the long-month descriptor is January–December.

%C  specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.

%d  specifies the day descriptor and uses 0 padding. The range for the day modifier is 01–31.

%e  specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.

%F  specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.

%j  specifies the day-of-year descriptor as a decimal number and uses a leading zero. The range for the day-of-year descriptor is 1–366.

%m  specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.

%m  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; logfile=nldate('24Feb2003'd,'%B-%d.log'); put logfile;</td>
<td>February-24.log</td>
</tr>
<tr>
<td>options locale=German_Germany; logfile=nldate('24Feb2003'd,'%B-%d.log'); put logfile;</td>
<td>Februar-24.log</td>
</tr>
</tbody>
</table>

The following example shows a weekday name that is created from a SAS date value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_unitedstates; weekname=nldate('24Feb2003'd,'%A'); put weekname;</td>
<td>Monday</td>
</tr>
<tr>
<td>options locale=German_Germany; weekname=nldate('24Feb2003'd,'%A'); put weekname;</td>
<td>Montag</td>
</tr>
</tbody>
</table>

See Also
Format:

- “NLDATE Format” on page 107
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 281.

**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.

%e

specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.
%F
specifies the long-weekday descriptor and uses blank padding. The range for the
day descriptor is Monday–Sunday.

%H
specifies the hour descriptor that is based on a 24-hour clock. The range for the
hour descriptor is 00–23.

%I
specifies the hour descriptor that is based on a 12-hour clock. The range for the
hour descriptor is 01–12.

%j
specifies the day-of-year descriptor as a decimal number and uses a leading zero.
The range for the day-of-year descriptor is 1–366.

%m
specifies the month descriptor and uses 0 padding. The range for the month
descriptor is 01–12.

%M
specifies the minute descriptor. The range for the minute descriptor is 00–59.

%o
specifies the month descriptor and uses blank padding. The range for the month
descriptor is 1–12.

%p
specifies a.m. or p.m. descriptor.

%S
specifies the second descriptor. The range for the second descriptor is 00–59.

%u
specifies the weekday descriptor as a number in the range of 1–7 that represents
Monday–Sunday.

%U
specifies the week-number-of-year descriptor by calculating the descriptor value
as the SAS date value and uses the number-of-week value within the year
(Sunday is considered the first day of the week). The number-of-week value is
represented as a decimal number in the range 0–53. A leading zero and a
maximum value of 53 is used.

%V
specifies the week-number-of-year descriptor by calculating the descriptor value
as the SAS date value. The number-of-week value is represented as a decimal
number in the range 01–53. A leading zero and a maximum value of 53 is used.
Weeks begin on a Monday and week 1 of the year is the week that includes both
January 4 and the first Thursday of the year. If the first Monday of January is the
2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding
year.

%W
specifies the weekday descriptor as a number in the range of 0–6 that represents
Sunday–Saturday.

%W
specifies the week-number-of-year descriptor by calculating the descriptor value
as SAS date value using the number of week within the year (Monday is
considered the first day of the week). The number-of-week value is represented
as a decimal number in the range of 0–53. A leading zero and a maximum value of 53 are used.

%y
 specifies the year (2-digit) descriptor. The range for the year descriptor is 00–99.

%Y
 specifies the year (4-digit) descriptor. The range for the year descriptor is 1970–2069.

Details
The NLDATM function converts the SAS datetime value to the datetime value of the specified locale by using the datetime descriptors.

Example
The following example shows a time (a.m or p.m.) that is created from a SAS datetime value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English;</td>
<td></td>
</tr>
<tr>
<td>time_ampm=nldatm('24Feb2003:12:39:43'dt,'%I%p');</td>
<td>12PM</td>
</tr>
<tr>
<td>put time_ampm;</td>
<td></td>
</tr>
<tr>
<td>options locale=German;</td>
<td></td>
</tr>
<tr>
<td>time_ampm=nldatm('24Feb2003:12:39:43'dt,'%I%p');</td>
<td>12nachm</td>
</tr>
<tr>
<td>put time_ampm;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLDATM Format” on page 128

NLTIME Function
Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME 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 281.</td>
</tr>
</tbody>
</table>

Syntax

NLTIME(time|datetime,descriptor,startpos)
**Required Arguments**

**time**
- specifies a SAS time value.

**datetime**
- specifies a SAS datetime value.

**descriptor**
- is a variable, or expression, that specifies the value of a descriptor. You can enter the following descriptors in uppercase or lowercase:
  
  - `#`
    - removes the leading zero from the result.
  
  - `%%`
    - specifies the % character.
  
  - `%H`
    - specifies the hour descriptor that is based on a 24-hour clock. The range for the hour descriptor is 00–23.
  
  - `%I`
    - specifies the hour descriptor that is based on a 12-hour clock. The range for the hour descriptor is 01–12.
  
  - `%M`
    - specifies the minute modifier. The range for the minute descriptor is 00–59.
  
  - `%P`
    - specifies the a.m. or p.m. descriptor.
  
  - `%S`
    - specifies the second descriptor. The range for the second descriptor is 00–59.

**startpos**
- is an integer that specifies the position at which the search should start and that specifies the direction of the search.

**Details**

The NLTIME function converts a SAS time or datetime value to the time value of the specified locale by using the time descriptors.

**Example**

The following example shows an a.m. or p.m. time that is created from a SAS time.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English;</td>
<td>12 PM</td>
</tr>
<tr>
<td>time_ampm=nltime('12:39:43't,'%i%p');</td>
<td></td>
</tr>
<tr>
<td>put time_ampm;</td>
<td></td>
</tr>
</tbody>
</table>
Statements | Results
---|---
options locale=German; | 12 nachm

time_ampm=nlt ime('12:39:43' t, '%i%p');

put time_ampm;

See Also

Format:
- “NLTIME Format” on page 242

**SASMSG Function**

Specifies a message from a data set. The returned message is based on the current locale and a specified key.

**Category:** Locale

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 281.

**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></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:

```sas
%macro demo_sasmsg;
  data _null_;  
  msg = sasmsg("nls.mymsg","IN_APW_SAVE_OK","noquote");  
  put msg=;  
  run;  
%mend demo_sasmsg;
```

**SAS Statements** | **Results**
--- | ---
options locale = en_US;  | msg=The Access Control key was successfully saved.  
%demo_sasmsg;  |  

options locale = es_ES;  | msg=La clave de control de acceso se ha guardado.  
%demo_sasmsg;  |  

options locale = french_France;  | msg=La clé de contrôle d'accès a bien été enregistrée.  
%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;
```

**SAS Statements** | **Results**
--- | ---
%PRT(en_US,&TABLEID,IN_EDIT)  | "Edit"  
%PRT(es_ES,&TABLEID,IN_EDIT)  | "Editar"  
%PRT(fr_FR,&TABLEID,IN_EDIT)  | "Modifier"
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 281.

**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 'uxxxxx, 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>
</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</td>
<td>msg= My cat. Your dog.</td>
</tr>
<tr>
<td>(&quot;nls.mymsg&quot;,&quot;IN_CD_LOG&quot;,&quot;en_US&quot;,&quot;N&quot;, &quot;cat&quot;,&quot;dog&quot;);</td>
<td></td>
</tr>
<tr>
<td>IN_CD_LOGINFO = My #$1s. Your #$2s</td>
<td></td>
</tr>
</tbody>
</table>

| IN_CD_LOGINFO = My #$2s. Your #$1s | msg= My dog. Your cat.      |

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 %NRQUOTE 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>&quot;Edit&quot;</td>
</tr>
<tr>
<td>%PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, es_ES));</td>
<td>&quot;Editar&quot;</td>
</tr>
<tr>
<td>%PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, fr));</td>
<td>&quot;Modifier&quot;</td>
</tr>
</tbody>
</table>

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 281.

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 19.1 on page 599 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;as&quot; &lt; &quot;às&quot; &lt; &quot;at&quot;).</td>
<td>Other differences between letters can also be considered secondary differences, depending on the language. A secondary difference is ignored when there is a primary difference anywhere in the strings.</td>
</tr>
<tr>
<td>TERTIARY or T</td>
<td>Upper and lowercase differences in characters are distinguished at the tertiary level (for example, &quot;ao&quot; &lt; &quot;Ao&quot; &lt; &quot;aò&quot;).</td>
<td>An example is the difference between large and small Kana. A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings.</td>
</tr>
<tr>
<td>QUATERNARY or Q</td>
<td>When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, &quot;ab&quot; &lt; &quot;a-b&quot; &lt; &quot;aB&quot;).</td>
<td>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.</td>
</tr>
<tr>
<td>IDENTICAL or I</td>
<td>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.</td>
<td>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.</td>
</tr>
</tbody>
</table>

case order

sorts uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL. The following table provides the values and information for the case order argument.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER or U</td>
<td>Sorts uppercase letters first, then the lowercase letters.</td>
</tr>
</tbody>
</table>
**LOWER or L**  
Sorts lowercase letters first, then the uppercase letters.

---

**numeric order**

orders numbers by the numeric value instead of the number's characters.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC or N</td>
<td>Order numbers (integers) by the numeric value. For example, &quot;8 Main St.&quot; would sort before &quot;45 Main St.&quot;.</td>
</tr>
</tbody>
</table>

---

**collation order**

There are two types of collation values: Phonebook and Traditional. If you do not select a collation value, then the user's locale-default collation is selected. The following table provides more information.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHONEBOOK or P</td>
<td>specifies a phonebook style ordering of characters. Select PHONEBOOK only with the German language.</td>
</tr>
<tr>
<td>TRADITIONAL or T</td>
<td>specifies a traditional style ordering of characters. Select TRADITIONAL only with the Spanish language.</td>
</tr>
</tbody>
</table>

---

**Details**

The SORTKEY function creates a linguistic sort key for data. You must enter at least one argument. If the length of the variable that receives the key is not large enough, the data truncates, and a warning is displayed.

**locale**

Locale values use the POSIX name (ll_RR). LL represents the two-letter language code, and RR represents the two-letter region code. For example, en_US is the POSIX name for English, United States. en represents the English language, and US represents the United States. If a locale value is not specified, then the session locale is used.

**strength**

The strength argument determines whether accents or case affect collating or matching text. If no value is specified for strength, then the locale determines the value. The following values can be specified for strength.

**PRIMARY**

This value includes base letters. An example is the letters, A, a, and Å are all processed the same.

**SECONDARY**

This value processes data the same as PRIMARY, and accents are processed. The letters A and a are processed equally, and Å is processed as an accented character.

**TERTIARY**

This value processes data the same as SECONDARY, and the character's case is processed. For example, A, a, and Å are all processed differently.
QUATERNARY
   This value processes data the same as TERTIARY, and punctuation is processed.

IDENTICAL
   This value process data the same as QUATERNARY, and code point is processed.

case order
   specifies to sort data using uppercase or lowercase letter. The following table shows examples of specifying the UPPER value or the LOWER value.

<table>
<thead>
<tr>
<th>UPPER</th>
<th>LOWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aztec</td>
<td>aztec</td>
</tr>
<tr>
<td>aztec</td>
<td>Aztec</td>
</tr>
<tr>
<td>Mars</td>
<td>mars</td>
</tr>
<tr>
<td>mars</td>
<td>Mars</td>
</tr>
</tbody>
</table>

collation order
   The collation order value PHONEBOOK is ignored unless the locale is a German language.
   The collation order value TRADITIONAL is ignored unless the locale is a Spanish language.
   A warning message is displayed for other locales.

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 281. |

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>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------</td>
<td>------</td>
<td>----------------------</td>
</tr>
<tr>
<td>FORMATNAME_NUMERIC</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_PERCENT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SANSSERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_CURSIVE</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_FANTASY</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_MONOSPACIAL</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BRUSH</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SWISS</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ITALIC</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DATE_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_AMPM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME_AMPM_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>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_YUVM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YUVM_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YUVQ_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YUVQ_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YUVW_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YUVW_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>ABMON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY7</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>DAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>AM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>PM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>INT_CURRENCY_SYMBOL</td>
<td>3</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>CURRENCY_SYMBOL</td>
<td>32</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_DECIMAL_POINT</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_THOUSANDS_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>MON_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_POSITION</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_SIGN_POSITION</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>DECIMAL_POINT</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>THOUSANDS_SEP</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>NEGATIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SIGN_POSITION</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SIGN_POSITION</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WIDTH</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Examples

Example 1
In the following locale example, the SETLOCALE function specifies the locale Japanese (jp_JP). The SETLOCALE function returns the previous locale. In this example, the previous locale was English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x=setlocale('ja_JP');</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td>x=English_UnitedStates</td>
</tr>
</tbody>
</table>

Example 2
In the following example, the SETLOCALE function returns the locale name where the element values are being changed:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>x=Japanese_Japan</td>
</tr>
<tr>
<td>x = setlocale(&quot;LC_MONETARY&quot;, 'zh_CN');</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>run;</td>
</tr>
<tr>
<td></td>
<td>x=Japanese_Japan</td>
</tr>
</tbody>
</table>

Example 3
In the following example, the SETLOCALE function changes the value of the specified key, DATE_YEAR_FORMAT:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data _null;</td>
<td>x=%Y¥</td>
</tr>
<tr>
<td>x=setlocale('DATE_YEAR_FORMAT', '¥%Y');</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>run;</td>
</tr>
<tr>
<td></td>
<td>x=%Y¥</td>
</tr>
</tbody>
</table>

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 281. 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 timezone ID. If you specify the timezone ID, the function returns the timezone ID if it is valid or returns a blank value if the ID is invalid.

Example
In the first example, the TIMEZONE option is set to JST. In the second example, TIMEZONE is set to a blank value. In the third example TIMEZONE is set to user-specified time zone. In the fourth example a valid timezoneid and an invalid timezoneid is displayed.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone=jst;</td>
<td>tzoneid=ASIA/TOKYO</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzid=tzoneid() ;</td>
<td></td>
</tr>
<tr>
<td>put tzid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options timezone='';</td>
<td>tzoneid=</td>
</tr>
<tr>
<td>data <em>null</em> ;</td>
<td></td>
</tr>
<tr>
<td>tzid=tzoneid() ;</td>
<td></td>
</tr>
<tr>
<td>put tzid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options timezone='xxx-12';</td>
<td>tzoneid=</td>
</tr>
<tr>
<td>/* user defined timezone */</td>
<td></td>
</tr>
<tr>
<td>data <em>null</em> ;</td>
<td></td>
</tr>
<tr>
<td>tzid=tzoneid() ;</td>
<td></td>
</tr>
<tr>
<td>put tzid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>data null;</td>
<td>name_valid=ASIA/TOKYO</td>
</tr>
<tr>
<td>name_valid=tzoneid('asia/tokyo');</td>
<td>name_invalid=</td>
</tr>
<tr>
<td>name_invalid=tzoneid('Milky Way');</td>
<td></td>
</tr>
<tr>
<td>put name_valid =;</td>
<td></td>
</tr>
<tr>
<td>put name_invalid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></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 281.
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 1, “Time Zone IDs and Time Zone Names,” on page 625.

**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>
<tbody>
<tr>
<td>options tz='';</td>
<td>tzname=</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzone=tzonename() ;</td>
<td></td>
</tr>
<tr>
<td>put tzone =;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
## 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 281](#).

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 1, “Time Zone IDs and Time Zone Names,” on page 625](#).

- **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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options tz='jst'; data <em>null</em>;</td>
<td>tzname=JST</td>
</tr>
<tr>
<td>tzname=tzonename() ; put tzname =; run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>options tz='xxx-12';</td>
<td>tzname=XXX</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzname=tzonename() ; put tzname =; run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>options tz='American/Chicago';</td>
<td>tzname=CDT</td>
</tr>
<tr>
<td>data_null_;</td>
<td></td>
</tr>
<tr>
<td>tzname=tzonename('01SEP2014:01:01:01'dt);</td>
<td></td>
</tr>
<tr>
<td>put tzname =;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>option TIMEZONE='AUSTRALIA/MELBOURNE';</code></td>
<td>39600</td>
</tr>
<tr>
<td><code>%PUT %SYSFUNC(TZONEOFF());</code></td>
<td></td>
</tr>
<tr>
<td><code>option TIMEZONE='AUSTRALIA/MELBOURNE';</code></td>
<td>3600</td>
</tr>
<tr>
<td><code>%PUT %SYSFUNC(TZONEOFF('EUROPE/ROME'));</code></td>
<td></td>
</tr>
<tr>
<td><code>data _null_;</code></td>
<td></td>
</tr>
<tr>
<td><code>dt1='05DEC2012:08:17:52'dt;</code></td>
<td></td>
</tr>
<tr>
<td><code>dt2='05JUN2012:08:17:52'dt;</code></td>
<td></td>
</tr>
<tr>
<td><code>offset1= TZONEOFF('EUROPE/MOSCOW', dt1);</code></td>
<td></td>
</tr>
<tr>
<td><code>offset2= TZONEOFF('EUROPE/MOSCOW', dt2);</code></td>
<td></td>
</tr>
<tr>
<td><code>put offset1= / offset2=;</code></td>
<td></td>
</tr>
<tr>
<td><code>run;</code></td>
<td></td>
</tr>
<tr>
<td><code>offset1=10800</code></td>
<td></td>
</tr>
<tr>
<td><code>offset2=14400</code></td>
<td></td>
</tr>
<tr>
<td><code>option TIMEZONE='EUROPE/MOSCOW';</code></td>
<td></td>
</tr>
<tr>
<td><code>data _null_;</code></td>
<td></td>
</tr>
<tr>
<td><code>dt1='05DEC2012:08:17:52'dt;</code></td>
<td></td>
</tr>
<tr>
<td><code>dt2='05JUN2012:08:17:52'dt;</code></td>
<td></td>
</tr>
<tr>
<td><code>offset1= TZONEOFF(dt1);</code></td>
<td></td>
</tr>
<tr>
<td><code>offset2= TZONEOFF(dt2);</code></td>
<td></td>
</tr>
<tr>
<td><code>put offset1= / offset2=;</code></td>
<td></td>
</tr>
<tr>
<td><code>run;</code></td>
<td></td>
</tr>
<tr>
<td><code>offset1=10800</code></td>
<td></td>
</tr>
<tr>
<td><code>offset2=14400</code></td>
<td></td>
</tr>
</tbody>
</table>

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 281.
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 1, “Time Zone IDs and Time Zone Names,” on page 625

**Details**

The TZONES2U() function returns UTC-based time for the specified TIMEZONE. The TZONES2U(time-zone-id) function with the time zone ID argument returns UTC-based time for the specified time zone ID. If the time zone name is not valid for the current locale, you receive an error.

**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 281.

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></td>
</tr>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>dstname=tzonedstname();</td>
<td>dstname=CDT</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 281.

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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone='America/Chicago';</td>
<td>dstoff=18000</td>
</tr>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>dstoff=tzonedstoff();</td>
<td></td>
</tr>
<tr>
<td>put dstoff;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

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 281.

This function is not valid in the CAS server.

Syntax

TZONESTTNAME()

TZONESTTNAME<time-zone id>

Optional Argument

time-zone-id

specifies a region or area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details

If the TZID is not specified, then the TIMEZONE ID that is specified by the TIMEZONE= option is used.
Example
The following example specifies the Asia/Osaka time zone.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone='Asia/Osaka'; data null; name=tzonesttname(); put name;</td>
<td>JST</td>
</tr>
<tr>
<td>run;</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 281.
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.
### Statements

```sas
options timezone='Asia/Osaka';
data null;
    name=tzonesttoff();
    put name;
rn;
```

<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 281. This function is not valid in the CAS server.

#### Syntax

```
TZONEU2S <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.
Statements

```sas
option locale=fr_FR TZ='AMERICA/DENVER';
data _null_
  utc_date = '2012-09-02T02:34:56+00:00';
  udt = input(utc_date,is8601dz.);
  sdt1 = tzzoneu2s(udt);
  sdt2 = tzzoneu2s(udt,'EUROPE/AMSTERDAM');
  sdt3 = tzzoneu2s(udt,'CET');
  put sdt1= datetime. / sdt2= datetime. / sdt3= datetime.;
run;
```

Results

<table>
<thead>
<tr>
<th></th>
<th>sdt1=</th>
<th>sdt2=</th>
<th>sdt3=</th>
</tr>
</thead>
<tbody>
<tr>
<td>utc_date</td>
<td>01SEP12:20:34:56</td>
<td>02SEP12:03:34:56</td>
<td>02SEP12:03:34:56</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 281.

**Syntax**

```
TRANTAB(string, trantab_name)
```

**Required Arguments**

- `string` input data that is transcoded.
- `trantab_name` translation table.

**Details**

The TRANTAB function transcodes a data string by using a translation table to remap the characters from one internal representation to another. The encoding of the data in the input string must match the encoding of table 1 in the translation table. The TRANTAB function remaps the data from the encoding using table 1.

Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= and ENCODING= system options as an improvement on direct use of translation tables. In SAS 9 translation tables are used only for transcoding external files. There is direct transcoding in SAS files, between the session encodings. SAS 9.2 supports the TRANTAB function for backward compatibility.

The LOCALE= system option is preferred in later SAS releases.

**CAUTION:**

Only experienced SAS users should use the TRANTAB function.
Example

The following example uses a translation table that transcodes data that is encoded in Latin2 to an uppercase Latin2 encoding:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>teststrg=trantab('testing','lat2_ucs'); put teststrg;</td>
<td>TESTING</td>
</tr>
</tbody>
</table>

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 281.

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

**Syntax**

STR=UNICODE(<instr> (<,Unicode type> ))

**Required Arguments**

*str*

Data string that has been converted to the current SAS session 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**  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:

```sas
/* Run this program using any CJK LOCALE */
data _null_;  
str1=unicode("\u0041\u0042\u0043");put str1=;          /* default */
str2=unicode("\u0041\u0042\u0043", "esc");put str2=;       /* ESC - Unicode Escape */
str3=unicode("\&#177;", "ncr");put str3=;       /* NCR - Numeric Character Representation */
str4=unicode("\&#22823;", "ncr");put str4=;       /* NCR - Numeric Character Representation */
str5=unicode('\u0061\u0062','paren');put str5=;       /* PAREN - Unicode Parenthesis Escape */
str6=unicode('2759'x,'ucs2');put str6=;       /* UCS2 - UCS2 encoding */
str7=unicode('5927'x,'ucs2b');put str7=;       /* UCS2B - UCS2 encoding with big endian */
str8=unicode('2759'x,'ucs2l');put str8=;       /* UCS2L - UCS2 encoding with little endian */
str9=unicode('27590000'x,'ucs4');put str9=;       /* UCS4 - UCS4 encoding */
str10=unicode('00005927'x,'ucs4B');put str10=;     /* UCS4B - UCS4 encoding with big endian */
str11=unicode('27590000'x,'ucs4l');put str11=;     /* UCS4L - UCS4 encoding with little endian */
str12=unicode('ESAA7'e,'utf8');put str12=;       /* UTF8 - UTF8 encoding */
str13=unicode('2759'x,'utf16');put str13=;       /* UTF16 - UTF16 encoding */
str14=unicode('5927'x,'utf16b');put str14=;       /* UTF16B - UTF16 encoding with big endian */
str15=unicode('2759'x,'utf16l');put str15=;       /* UTF16L - UTF16 encoding with little endian */
str16=unicode('\u000020bb7','esc');put str16=;     /* ESC - Unicode Escape for Supplementary Character */
str17=unicode("\&#134071;", "ncr");put str17=;     /* NCR - Numeric Character Representation for Supplementary Character */
str18=unicode('\u000020bb7','paren');put str18=;     /* PAREN - Unicode Parenthesis Escape for Supplementary Character */
run;
```

Here are the results from the UNICODE function example:

```
str1=ABC
str2=ABC
str3=�
str4=¥
str5=¥
str6=¥
str7=¥
str8=¥
str9=¥
str10=¥
str11=¥
str12=¥
str13=¥
str14=¥
str15=¥
str16=¥
str17=¥
str18=¥
```

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 281.

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

Syntax

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:
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 281.

### 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><code>len1=unicodelen(&quot;abcΛ&quot;);</code></td>
<td>len1=4</td>
</tr>
<tr>
<td><code>len2=unicodelen(&quot;\u0041\u0042\u0043\u5927&quot;,&quot;esc&quot;);</code></td>
<td>len2=4</td>
</tr>
<tr>
<td><code>len3=unicodelen(&quot;&amp;#22823;&quot;,&quot;ncr&quot;);</code></td>
<td>len3=1</td>
</tr>
<tr>
<td><code>len4=unicodelen(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;,&quot;paren&quot;);</code></td>
<td>len4=2</td>
</tr>
</tbody>
</table>

### See Also

**Functions:**
- “UNICODEWIDTH Function” on page 372
UNICODEWIDTH Function

Specifies the length of a display unit for the Unicode data.

**Category:** Character

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 281.

**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>
<tbody>
<tr>
<td>len1=unicodewidth(&quot;abc 冬&quot;);</td>
<td>len1=5</td>
</tr>
<tr>
<td>len2=unicodewidth(&quot;\u0041\u0042\u0043\u5927&quot;,'esc');</td>
<td>len2=5</td>
</tr>
<tr>
<td>len3=unicodewidth(&quot;大 &quot;,'ncr');</td>
<td>len3=2</td>
</tr>
<tr>
<td>len4=unicodewidth(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;,'paren');</td>
<td>len4=2</td>
</tr>
</tbody>
</table>

**See Also**

**Functions:**

- “UNICODELEN Function” on page 371

VARTRANSCODE Function

Returns the transcode attribute of a SAS data set variable.

**Category:** Variable Information
Syntax

VARTRANSCODE(data-set-id, var-num)

Required Arguments

data-set-id
  specifies the data set identifier that the OPEN function returns.

var-num
  specifies the position of the variable in the SAS data set.

Tip  The VARNUM function returns this value.

Details

Transcoding is the process of converting data from one encoding to another. The VARTRANSCODE function returns 0 if the var-num variable does not transcode its value, or 1 if the var-num variable transcodes its value.


Example

The following example shows how to determine whether a character variable is transcoded:

data a;
  attrib x length=$3. transcode=no;
  attrib y length=$3. transcode=yes;
  x='abc';
  y='xyz';
run;
data _null_
  dsid=open('work.a','i');
  nobs=attrn(dsid,"nobs");
  nvars=attrn(dsid,"nvars");
  do i=1 to nobs;
    xrc=fetch(dsid,1);
    do j=1 to nvars;
      transcode = vartranscode(dsid,j);
      put transcode=;
    end;
  end;
run;

SAS writes the following output to the log:

transcode=0
transcode=1
See Also

Functions:

- “ATTRN Function” in SAS 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 398
- “VTRANSCODEX Function” on page 399
## SAS Functions for CAS and SAS by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>BASECHAR Function (p. 376)</td>
<td>Converts characters to base characters.</td>
</tr>
<tr>
<td></td>
<td>KSTRIP Function (p. 389)</td>
<td>Removes leading and trailing blanks from a character string.</td>
</tr>
<tr>
<td>DBCS</td>
<td>KCOMPRESS Function (p. 378)</td>
<td>Removes specified characters from a character expression.</td>
</tr>
<tr>
<td></td>
<td>KCOUNT Function (p. 379)</td>
<td>Returns the number of double-byte characters in an 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>KINDEX Function</td>
<td>(p. 380)</td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td>KINDEXC Function</td>
<td>(p. 381)</td>
<td>Searches a character expression for specified characters.</td>
</tr>
<tr>
<td>KLEFT Function</td>
<td>(p. 382)</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td>KLENGTH Function</td>
<td>(p. 383)</td>
<td>Returns the length of an argument.</td>
</tr>
<tr>
<td>KLOWCASE Function</td>
<td>(p. 384)</td>
<td>Converts all letters in an argument to lowercase.</td>
</tr>
<tr>
<td>KREVERSE Function</td>
<td>(p. 385)</td>
<td>Reverses a character expression.</td>
</tr>
<tr>
<td>KRIGHT Function</td>
<td>(p. 386)</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td>KSCAN Function</td>
<td>(p. 387)</td>
<td>Selects a specified word from a character expression.</td>
</tr>
<tr>
<td>KSTRCAT Function</td>
<td>(p. 388)</td>
<td>Concatenates two or more character expressions.</td>
</tr>
<tr>
<td>KSUBSTR Function</td>
<td>(p. 390)</td>
<td>Extracts a substring from an argument.</td>
</tr>
<tr>
<td>KTRIM Function</td>
<td>(p. 392)</td>
<td>Removes trailing DBCS blanks and SO/SI from character expressions.</td>
</tr>
<tr>
<td>KUPCASE Function</td>
<td>(p. 393)</td>
<td>Converts all letters in an argument to uppercase.</td>
</tr>
<tr>
<td>KUPDATE Function</td>
<td>(p. 394)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td>KUPDATES Function</td>
<td>(p. 395)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
</tr>
<tr>
<td>KVERIFY Function</td>
<td>(p. 397)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
</tr>
<tr>
<td>Variable Information</td>
<td></td>
<td>VTRANSCODE Function (p. 398) Returns a value that indicates whether transcoding is enabled for the specified character variable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VTRANSCODEX Function (p. 399) Returns a value that indicates whether transcoding is enabled for the specified argument.</td>
</tr>
</tbody>
</table>

**Dictionary**

**BASECHAR Function**

Converts characters to base characters.

**Category:** Character
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 281.

Syntax

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

Required Arguments

\( \text{str} \)

data string that is converted.

\( \text{instr} \)
input data string.

\( \text{Unicode type} \)
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 (Å).
- 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>Note: The SAS session encoding is UTF-8. data cities; length name$24 name_utf8$24; input name; name_utf8 = basechar(name); put name_utf8=; cards; Mühlenfließ LaUna 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 ZielonaGóra</td>
</tr>
</tbody>
</table>
KCOMPRESS Function

Removes specified characters from a character 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 281.

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

**Syntax**

\[
\text{KCOMPRESS}(\text{source}, <\text{characters-to-remove}>)
\]

**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.expression.

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

See Also

Functions:
- “KLEFT Function” on page 382
- “KTRIM Function” on page 392

KCOUNT Function

Returns the number of double-byte characters in 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 281.

Syntax

KCOUNT(source)

Required Argument

source
 specifies the character expression to count.

Details

See “Internationalization Compatibility for SAS String Functions” on page 281 for restrictions and more information.

Example

The following example uses Japanese characters.
KINDEX 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 281](#).

**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 281 for restrictions and more information.

The KINDEX function searches source, from left to right, for the first occurrence of the string that is specified in excerpt, and returns the position in source of the string’s first character. If the string is not found in source, KINDEX returns a value of 0. If there are multiple occurrences of the string, KINDEX returns only the position of the first occurrence.

### Example

The following example uses Japanese characters.

---

```plaintext
data _null_;  
  text='abcあいう';  
  result=kcount(text);  
  put result=;  
run;
```
KINDEXC 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 281.

**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 281 for restrictions and more information.

---

Statements | Results
---|---
data _null_;  
text=`漢字文字列の検索';  
result=kindex(text,'検索');  
put result=;  
text2=`探す';  
result=kindex(text,text2);  
put result=;  
run;

result=7
result=0
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.

```
data _null_;   result=9
  a='A B C.def (漢字123456)';
  result=kindexc(a,'1234567890','感漢');
  put result=;
run;
```

### See Also

Function:
- “KINDEX Function” on page 380

### KLEFT Function

Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.

**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 281.  
**Tip:** Non-DBCS equivalent function is LEFT in SAS Viya Functions and CALL Routines: Reference.

### Syntax

\[
\text{KLEFT}(\text{argument})
\]

**Required Argument**

- `argument` specifies any SAS character expression.
Details

See “Internationalization Compatibility for SAS String Functions” on page 281 for restrictions and more information.

KLEFT returns an argument and removes the leading blanks.

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>a=’漢字 的 テスト’;</td>
<td>1---------</td>
</tr>
<tr>
<td>b=kleft(a);</td>
<td>2--------+</td>
</tr>
<tr>
<td>put ’----------1-------2-----’;</td>
<td>漢字 的 テスト</td>
</tr>
<tr>
<td>put a $25.;</td>
<td>漢字 的 テスト</td>
</tr>
<tr>
<td>put b $25.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:
- “KCOMPRESS Function” on page 378
- “KRIGHT Function” on page 386
- “KTRIM Function” on page 392

KLENGTH Function

Returns the length of an argument.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBSCS (UTF8). For more information, see Internationalization Compatibility on page 281.

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

Syntax

KLENGTH(argument)

Required Argument

argument specifies any SAS expression.
Details

See “Internationalization Compatibility for SAS String Functions” on page 281 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.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=6</td>
</tr>
<tr>
<td>text='abcあいう';</td>
<td></td>
</tr>
<tr>
<td>result=klength(text);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KLOWCASE Function

Converts all letters in an argument to lowercase.

- **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 281.
- **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 281 for restrictions and more information.

The KLOWCASE function copies a character argument, converts all uppercase letters to lowercase letters, and returns the altered value as a 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>result=abcあいうえお</td>
</tr>
<tr>
<td>result=klowcase('ABCあいうえお');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KREVERSE Function

Reverses a character 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 281.

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 281 for restrictions and more information.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=トステの字漢</td>
</tr>
<tr>
<td>result=kreverse('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.

**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 281.

**Tip:** See “RIGHT Function” in SAS Viya Functions and CALL Routines: Reference.

**Syntax**

```sas
KRIGHT(argument)
```

**Required Argument**

`argument` specifies any SAS character expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 281 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.

**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>a='漢字のテスト';</td>
<td>漢字のテスト</td>
</tr>
<tr>
<td>b=kright(a);</td>
<td>漢字のテスト</td>
</tr>
<tr>
<td>put '------------1-----------2-----';</td>
<td></td>
</tr>
<tr>
<td>put a $25.;</td>
<td></td>
</tr>
<tr>
<td>put b $25.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Functions:

- “KCOMPRESS Function” on page 378
- “KLEFT Function” on page 382
- “KTRIM Function” on page 392
KSCAN Function

Selects a specified word from a character 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 281.*

**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 281 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.
Statements

```
data _null_;  
   length x $20 y $20;  
   text1='これは漢字関数のテストです。';  
   x='norblank';  
   i=1;  
   do until(x='');  
      x=kscan(text1,i,'はの');  
      put x=;  
      i=i+1;  
   end;  
   y='norblank';  
   i=-1;  
   do until(y='');  
      y=kscan(text1,i,'はの');  
      put y=;  
      i=i-1;  
   end;  
run;```

Results

```
x=これ  
x=漢字関数  
x=テストです。  
x=  
y=テストです。  
y=漢字関数  
y=これ  
y=```

### KSTRCAT Function

Concatenates two or more 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 281.
- **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 281 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.

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 281.

Syntax

\[ \text{KSTRIP}(string) \]

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.

KSTRIP and STRIP functions are used for data normalization purposes. The KSTRIP function should be used in a DBCS environment. The KSTRIP and STRIP functions have similar performance actions.

**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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>93B893B820</td>
</tr>
<tr>
<td>length a $5;</td>
<td></td>
</tr>
<tr>
<td>v = unicode('\u0020\u0020\u8404\u8404\u8404\u8404\u0020\u0020');</td>
<td></td>
</tr>
<tr>
<td>put v = hex.;</td>
<td></td>
</tr>
<tr>
<td>a = kstrip(v);</td>
<td></td>
</tr>
<tr>
<td>put a = hex.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

**Function**

- “STRIP Function” in *SAS Viya Functions and CALL Routines: Reference*

**KSUBSTR Function**

Extracts a substring from an 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 281*.

**Tip:** See “SUBSTR (left of =) Function” in *SAS Viya Functions and CALL Routines: Reference*.

**Syntax**

$$\text{KSUBSTR}(\text{argument}, \text{position}<, n>)$$
Required Arguments

*argument*
specifies any SAS character expression.

*position*
specifies a numeric expression that is the beginning character position.

*n*
specifies a numeric expression that is the length of the substring to extract.

Interaction
If *n* is larger than the length of the expression that remains in *argument* after *position*, SAS extracts the remainder of the expression.

Tip
If you omit *n*, SAS extracts the remainder of the expression.

Details
See “Internationalization Compatibility for SAS String Functions” on page 281 for restrictions and more information.

The KSUBSTR function returns a portion of an expression that you specify in *argument*. The portion begins with the character specified by *position* and is the number of characters specified by *n*.

A variable that is created by KSUBSTR obtains its length from the length of *argument*.

Example
The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>text='漢字文字列の抽出';</td>
<td></td>
</tr>
<tr>
<td>result=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:

- “KSUBSTRB Function” on page 330
KTRIM Function
Removes trailing DBCS blanks and SO/SI from 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 281.

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 281 for restrictions and more information.

KTRIM copies a character argument, removes all trailing blanks, and returns the trimmed argument as a result. If the argument is blank, KTRIM returns one blank. KTRIM is useful for concatenating because concatenation does not remove trailing blanks.

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.

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>; part1='漢字の'; part2='テスト'; hasblank=part1</td>
<td></td>
</tr>
</tbody>
</table>
KUPCASE Function

Converts all letters in an argument to uppercase.

**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 281.

**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 281 for restrictions and more information.

The KUPCASE function copies a character argument, converts all lowercase letters to uppercase letters, and returns the altered value as a 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>result=kupcase('abcあいうえお');</td>
<td>result=ABCあいうえお</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
KUPDATE Function

Inserts, deletes, and replaces character value contents.

**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 281.

**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.

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

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 281 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>x1=1 56</td>
</tr>
<tr>
<td>str='123456';</td>
<td>y1=156</td>
</tr>
<tr>
<td>x1=str; substr(x1,2,3)=&quot;&quot;;</td>
<td>z1=1 56</td>
</tr>
<tr>
<td>y1=kupdate(str,2,3);</td>
<td>x2=1abc56</td>
</tr>
<tr>
<td>z1=kupdates(str,2,3);</td>
<td>y2=1abc56</td>
</tr>
<tr>
<td>put x1= / y1= / z1=;</td>
<td>z2=1abc56</td>
</tr>
<tr>
<td>x2=str; substr(x2,2,3)=&quot;abcd&quot;;</td>
<td>x3=1ab 56</td>
</tr>
<tr>
<td>y2=kupdate(str,2,3,&quot;abcd&quot;);</td>
<td>y3=1ab56</td>
</tr>
<tr>
<td>z2=kupdates(str,2,3,&quot;abcd&quot;);</td>
<td>z3=1ab 56</td>
</tr>
<tr>
<td>put x2= / y2= / z2=;</td>
<td></td>
</tr>
<tr>
<td>x3=str; substr(x3,2,3)=&quot;ab&quot;;</td>
<td></td>
</tr>
<tr>
<td>y3=kupdate(str,2,3,&quot;ab&quot;);</td>
<td></td>
</tr>
<tr>
<td>z3=kupdates(str,2,3,&quot;ab&quot;);</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>x1=漢字のテスト</td>
</tr>
<tr>
<td>x1=kupdate('感じのテスト',1,2,'漢字');</td>
<td>x2=kaのテスト</td>
</tr>
<tr>
<td>x2=kupdate(x1,1,2,'kanji');</td>
<td>x3=テスト</td>
</tr>
<tr>
<td>x3=kupdate(x1,1,3);</td>
<td>x4=漢字かんじ</td>
</tr>
<tr>
<td>x4=kupdate(x1,3,'かんじ');</td>
<td></td>
</tr>
<tr>
<td>put x1= / x2= / x3= / x4=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:
- “KUPDATES Function” on page 395
- “KUPDATEB Function” on page 332

KUPDATES Function
Inserts, deletes, and replaces character value contents.

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 281.

Syntax

KUPDATES(argument,position,n<,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 281 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.
The following example shows the difference between KUPDATE and KUPDATES.

### Data _null_;

```
data null;
  x1=kupdates('感じのテスト',1,2,'漢字'); /* x1=漢字のテスト */
  x2=kupdates(x1,1,2,'kanji'); /* x2=kaのテスト */
  x3=kupdates(x1,1,3);
  x4=kupdates(x1,3,'かんじ'); /* x4=漢字かんじ */
  put x1= / y1= / z1= / x2= / y2= / z2= / x3= / y3= / z3= / x4= / y4= / z4=;
run;
```

### Example 2

See Also

**Functions:**

- “KUPDATE Function” on page 394

### KVERIFY 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 281.

**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 281 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>

VTRANSCODE Function

Returns a value that indicates whether transcoding is enabled for the specified character variable.

Category: Variable Information

Syntax

VTRANSCODE (var)
**Required Argument**

*var*

specifies a character variable that is expressed as a scalar or as an array reference.

**Restriction**

You cannot use an expression as an argument.

**Details**

The VTRANSCODE function returns 0 if transcoding is off, and 1 if transcoding is on. By default, all character variables in the DATA step are transcoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

**Comparisons**

- The VTRANSCODE function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however, evaluates the argument to determine the variable name. The function then returns the transcoding status (on or off) that is associated with that variable name.

- The VTRANSCODE function does not accept an expression as an argument. The VTRANSCODEX function accepts expressions, but the value of the specified expression cannot denote an array reference.

- Related functions return the value of other variable attributes, such as the variable name, type, format, and length. For a list of the variable attributes, see the “Variable Information” functions in *SAS 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>rc1=0</td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td></td>
</tr>
<tr>
<td>rcl = vtranscode(y);</td>
<td></td>
</tr>
<tr>
<td>put rcl=;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Functions:**

- “VTRANSCODEX Function” on page 399

**Statements:**

- ATTRIB in

---

**VTRANSCODEX Function**

Returns a value that indicates whether transcoding is enabled for the specified argument.
Category: Variable Information

Syntax

VTRANSCODEX (var)

Required Argument

var

specifies any SAS character expression that evaluates to a character variable name.

Restriction  The value of the specified expression cannot denote an array reference.

Details

The VTRANSCODEX function returns 0 if transcoding is off, and 1 if transcoding is on.

By default, all character variables in the DATA step are transcoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

Comparisons

• The VTRANSCODE function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however, evaluates the argument to determine the variable name. The function then returns the transcoding status (on or off) that is associated with that variable name.

• The VTRANSCODE function does not accept an expression as an argument. The VTRANSCODEX function accepts expressions, but the value of the specified expression cannot denote an array reference.

• Related functions return the value of other variable attributes, such as the variable name, type, format, and length. For a list of the variable attributes, see the “Variable Information” functions in SAS 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; attrib y transcode = no; rcl = vtranscodex('y'); put rcl=;</td>
<td>rcl=0</td>
</tr>
</tbody>
</table>

See Also

Functions:

• “VTRANSCODE Function” on page 398

Statements:
• ATTRIB
Part 7

Informats for NLS

Chapter 15

Dictionary of Informats for NLS

405
404
Chapter 15
Dictionary of Informats for NLS

Informats by Category

Dictionary .......................................................... 414
SCPTDW Informat .................................................. 414
SCPTWD Informat .................................................. 415
EURO Informat .................................................... 416
EUROX Informat ................................................... 418
MINGUO Informat ................................................ 419
NENGO Informat .................................................. 421
NLDATE Informat ................................................ 422
NLDATEW Informat ............................................... 423
NLDATM Informat ............................................... 424
NLDATMAP Informat ............................................ 425
NLDATMW Informat ............................................ 426
NLMNIAED Informat ............................................. 427
NLMNIAUD Informat ............................................. 428
NLMNIBGN Informat ............................................ 429
NLMNIBRL Informat ............................................ 430
NLMNICAD Informat ........................................... 431
NLMNICHF Informat ........................................... 432
NLMNICNY Informat ........................................... 433
NLMNICZK Informat ........................................... 434
NLMNIDKK Informat ........................................... 435
NLMNIEEK Informat ........................................... 436
NLMNIEGP Informat ........................................... 437
NLMNIEUR Informat ........................................... 438
NLMNIGBP Informat ........................................... 439
NLMNIHKD Informat ........................................... 440
NLMNIHRK Informat ........................................... 441
NLMNIHUF Informat ........................................... 442
NLMNIIDR Informat ........................................... 443
NLMNIILS Informat ............................................ 444
NLMNIINR Informat ............................................ 445
NLMNIJPY Informat ............................................ 446
NLMNIKRW Informat ........................................... 447
NLMNILTL Informat ........................................... 448
NLMNILVL Informat ........................................... 449
NLMNIMOP Informat ........................................... 450
NLMNIMXN Informat ........................................... 451
NLMNIMYR Informat ........................................... 452
NLMNINOK Informat ........................................... 453
NLMNINZD Informat ........................................... 454
NLMIPLN Informat ........................................ 455
NLMINRUB Informat ....................................... 456
NLNISEK Informat ........................................ 457
NLNISGD Informat ........................................ 458
NLNITHB Informat ........................................ 459
NLNITRY Informat ........................................ 460
NLNITWD Informat ........................................ 461
NLNIUSD Informat ......................................... 462
NLNIZAR Informat ........................................ 463
NLNLAEED Informat ...................................... 464
NLNLAUD Informat ........................................ 465
NLNLBGN Informat ....................................... 466
NLNLBRL Informat ........................................ 467
NLNLCAD Informat ....................................... 468
NLNLCHF Informat ....................................... 469
NLNLCLR Informat ........................................ 470
NLNLCHF Informat ....................................... 471
NLNLNOK Informat ...................................... 472
NLNLNOK Informat ....................................... 473
NLNLNZD Informat ...................................... 474
NLNLNZD Informat ....................................... 475
NLNLGBP Informat ...................................... 476
NLNLHKD Informat ....................................... 477
NLNLHRK Informat ...................................... 478
NLNLHUF Informat ....................................... 479
NLNLIDR Informat ...................................... 480
NLNLILS Informat ....................................... 481
NLNLINR Informat ...................................... 482
NLNLJPY Informat ....................................... 483
NLNLKRW Informat ..................................... 484
NLNLKRW Informat ..................................... 485
NLNLKRW Informat ..................................... 486
NLNLKRW Informat ..................................... 487
NLNLKRW Informat ..................................... 488
NLNLKRW Informat ..................................... 489
NLNLKRW Informat ..................................... 490
NLNLKRW Informat ..................................... 491
NLNLKRW Informat ..................................... 492
NLNLKRW Informat ..................................... 493
NLNLKRW Informat ..................................... 494
NLNLKRW Informat ..................................... 495
NLNLKRW Informat ..................................... 496
NLNLKRW Informat ..................................... 497
NLNLKRW Informat ..................................... 498
NLNLKRW Informat ..................................... 499
NLNLKRW Informat ..................................... 500
NLNLKRW Informat ..................................... 501
NLNLKRW Informat ..................................... 502
NLNLKRW Informat ..................................... 503
NLNLKRW Informat ..................................... 504
NLNLKRW Informat ..................................... 505
NLNLKRW Informat ..................................... 506
NLNLKRW Informat ..................................... 507
NLNLKRW Informat ..................................... 508
NLNLKRW Informat ..................................... 509
NLNLKRW Informat ..................................... 510
NLNLKRW Informat ..................................... 511
NLNLKRW Informat ..................................... 512

Chapter 15  •  Dictionary of Informs for NLS
Informats by Category

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>$UCS2B Informat (p. 511)</td>
<td>Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$UCS2BE Format (p. 512)</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>$UCS2L Format (p. 513)</td>
<td>Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS2LE Format (p. 514)</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>$UCS2X Format (p. 515)</td>
<td>Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS2XE Format (p. 516)</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>$UCS4B Format (p. 517)</td>
<td>Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4L Format (p. 518)</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>$UCS4X Format (p. 519)</td>
<td>Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4XE Format (p. 521)</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>$UESC Format (p. 522)</td>
<td>Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UESCE Format (p. 523)</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>$UNCR Format (p. 524)</td>
<td>Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UNCRE Format (p. 525)</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 Format (p. 526)</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</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</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>UTF8X Informat</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>Date and Time</td>
<td>MINGUO Informat</td>
<td>Reads dates in Taiwanese format.</td>
</tr>
<tr>
<td></td>
<td>NENGO Informat</td>
<td>Reads Japanese date values in the form eyymmdd.</td>
</tr>
<tr>
<td></td>
<td>NLDATE Informat</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</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</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</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</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</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</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>Hebrew Text Handling</td>
<td>SCPTDW Informat</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>SCPTWD Informat</td>
<td>Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding.</td>
</tr>
<tr>
<td>Numeric</td>
<td>EURO Informat</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</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></td>
<td>NLMNIAED Informat (p. 427)</td>
<td>Reads the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>NLMNIAUD Informat (p. 428)</td>
<td>Reads the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIBGN Informat (p. 429)</td>
<td>Reads the monetary format of the international expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>NLMNIBRL Informat (p. 430)</td>
<td>Reads the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNICAD Informat (p. 431)</td>
<td>Reads the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNICCHF Informat (p. 432)</td>
<td>Reads the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNICNY Informat (p. 433)</td>
<td>Reads the monetary format of the international expression for China.</td>
</tr>
<tr>
<td></td>
<td>NLMNICZK Informat (p. 434)</td>
<td>Reads the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>NLMNIDKK Informat (p. 435)</td>
<td>Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>NLMNIEEEK Informat (p. 436)</td>
<td>Reads the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIEGP Informat (p. 437)</td>
<td>Reads the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>NLMNIEUR Informat (p. 438)</td>
<td>Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td></td>
<td>NLMNIGBP Informat (p. 439)</td>
<td>Reads the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td>NLMNIHKD Informat (p. 440)</td>
<td>Reads the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td></td>
<td>NLMNIHRK Informat (p. 441)</td>
<td>Reads the monetary format of the international expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIHUF Informat (p. 442)</td>
<td>Reads the monetary format of the international expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>NLMNIIDR Informat (p. 443)</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>NLMNIILS Informat (p. 444)</td>
<td>Reads the monetary format of the international expression for Israel.</td>
<td></td>
</tr>
<tr>
<td>NLMNIINR Informat (p. 445)</td>
<td>Reads the monetary format of the international expression for India.</td>
<td></td>
</tr>
<tr>
<td>NLMNIJPY Informat (p. 446)</td>
<td>Reads the monetary format of the international expression for Japan.</td>
<td></td>
</tr>
<tr>
<td>NLMNIKRW Informat (p. 447)</td>
<td>Reads the monetary format of the international expression for South Korea.</td>
<td></td>
</tr>
<tr>
<td>NLMNILTL Informat (p. 448)</td>
<td>Reads the monetary format of the international expression for Lithuania.</td>
<td></td>
</tr>
<tr>
<td>NLMNILVL Informat (p. 449)</td>
<td>Reads the monetary format of the international expression for Latvia.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMOP Informat (p. 450)</td>
<td>Reads the monetary format of the international expression for Macau.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMXN Informat (p. 451)</td>
<td>Reads the monetary format of the international expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>NLMNIMYR Informat (p. 452)</td>
<td>Reads the monetary format of the international expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>NLMNINOK Informat (p. 453)</td>
<td>Reads the monetary format of the international expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>NLMNINZD Informat (p. 454)</td>
<td>Reads the monetary format of the international expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>NLMNIPLN Informat (p. 455)</td>
<td>Reads the monetary format of the international expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>NLMNIRUB Informat (p. 456)</td>
<td>Reads the monetary format of the international expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>NLMNISEK Informat (p. 457)</td>
<td>Reads the monetary format of the international expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>NLMNISGD Informat (p. 458)</td>
<td>Reads the monetary format of the international expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>NLMNITHB Informat (p. 459)</td>
<td>Reads the monetary format of the international expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>NLMNITRY Informat (p. 460)</td>
<td>Reads the monetary format of the international expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>NLMNITWD Informat (p. 461)</td>
<td>Reads the monetary format of the international expression for Taiwan.</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>NLMNIUSD Informat (p. 462)</td>
<td>Reads the monetary format of the international expression for Puerto Rico and the United States.</td>
<td></td>
</tr>
<tr>
<td>NLMNIZAR Informat (p. 463)</td>
<td>Reads the monetary format of the international expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>NLMNLAED Informat (p. 464)</td>
<td>Reads the monetary format of the local expression for the United Arab Emirates.</td>
<td></td>
</tr>
<tr>
<td>NLMNLAUD Informat (p. 465)</td>
<td>Reads the monetary format of the local expression for Australia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLBGN Informat (p. 466)</td>
<td>Reads the monetary format of the local expression for Bulgaria.</td>
<td></td>
</tr>
<tr>
<td>NLMNLBRL Informat (p. 467)</td>
<td>Reads the monetary format of the local expression for Brazil.</td>
<td></td>
</tr>
<tr>
<td>NLMNLCAD Informat (p. 468)</td>
<td>Reads the monetary format of the local expression for Canada.</td>
<td></td>
</tr>
<tr>
<td>NLMNLCHF Informat (p. 469)</td>
<td>Reads the monetary format of the local expression for Liechtenstein and Switzerland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLCHYN Informat (p. 470)</td>
<td>Reads the monetary format of the local expression for China.</td>
<td></td>
</tr>
<tr>
<td>NLMNLČZK Informat (p. 471)</td>
<td>Reads the monetary format of the local expression for the Czech Republic.</td>
<td></td>
</tr>
<tr>
<td>NLMNLDKK Informat (p. 472)</td>
<td>Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLEEK Informat (p. 473)</td>
<td>Reads the monetary format of the local expression for Estonia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLEG Informat (p. 474)</td>
<td>Reads the monetary format of the local expression for Egypt.</td>
<td></td>
</tr>
<tr>
<td>NLMNLEUR Informat (p. 475)</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>
<td></td>
</tr>
<tr>
<td>NLMNLGBP Informat (p. 476)</td>
<td>Reads the monetary format of the local expression for the United Kingdom.</td>
<td></td>
</tr>
<tr>
<td>NLMNLHKD Informat (p. 477)</td>
<td>Reads the monetary format of the local expression for Hong Kong.</td>
<td></td>
</tr>
<tr>
<td>NLMNLRKR Informat (p. 478)</td>
<td>Reads the monetary format of the local expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLHUF Informat (p. 479)</td>
<td>Reads the monetary format of the local expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>NLMNLLIEDR Informat (p. 480)</td>
<td>Reads the monetary format of the local expression for Indonesia.</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>NLMNLILS Informat (p. 481)</td>
<td>Reads the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>NLMNLINR Informat (p. 482)</td>
<td>Reads the monetary format of the local expression for India.</td>
</tr>
<tr>
<td></td>
<td>NLMNLJPY Informat (p. 483)</td>
<td>Reads the monetary format of the local expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLKRW Informat (p. 484)</td>
<td>Reads the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLTL Informat (p. 485)</td>
<td>Reads the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>NLMNLLVL Informat (p. 486)</td>
<td>Reads the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMO Informat (p. 487)</td>
<td>Reads the monetary format of the local expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMXN Informat (p. 488)</td>
<td>Reads the monetary format of the local expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>NLMNLMYR Informat (p. 489)</td>
<td>Reads the monetary format of the local expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNOK Informat (p. 490)</td>
<td>Reads the monetary format of the local expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMNLNZD Informat (p. 491)</td>
<td>Reads the monetary format of the local expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNPLN Informat (p. 492)</td>
<td>Reads the monetary format of the local expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLRO Informat (p. 493)</td>
<td>Reads the monetary format of the local expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLSEK Informat (p. 494)</td>
<td>Reads the monetary format of the local expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>NLMNLSD Informat (p. 495)</td>
<td>Reads the monetary format of the local expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTTHB Informat (p. 496)</td>
<td>Reads the monetary format of the local expression for Thailand.</td>
</tr>
<tr>
<td></td>
<td>NLMNTRY Informat (p. 497)</td>
<td>Reads the monetary format of the local expression for Turkey.</td>
</tr>
<tr>
<td></td>
<td>NLMNLTWD Informat (p. 498)</td>
<td>Reads the monetary format of the local expression for Taiwan.</td>
</tr>
<tr>
<td></td>
<td>NLMNLUSD Informat (p. 499)</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. 500)</td>
<td>Reads the monetary format of the local expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>NLMNY Informat (p. 501)</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>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLMNYI Informat (p. 502)</td>
<td>Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLNUM Informat (p. 504)</td>
<td>Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLNUMI Informat (p. 505)</td>
<td>Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLPCT Informat (p. 506)</td>
<td>Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLPCTI Informat (p. 508)</td>
<td>Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
</tbody>
</table>

**Dictionary**

**$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**

$CPTDWw.$

**Syntax Description**

w specifies the width of the input field.

Default 200

Range 1–32000

**Comparisons**

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

**Example**

The following example uses the input value of 808182.
$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

$CPTWDw.

Syntax Description

w

specifies the width of the input field.

Default 200

Range 1–32000

Comparisons

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

Example

The following example uses the input value of ¹²³⁴.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('808182',$cptdw6.); put x;</td>
<td>¹²³⁴</td>
</tr>
</tbody>
</table>

See Also

Formats:

- “$CPTDW Format” on page 90
- “$CPTWD Format” on page 91

Informat:

- “$CPTWD Informat” on page 415
EURO Informat

Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.

**Category:** Numeric

### Syntax

`EUROw.d`

**Syntax Description**

- `w` specifies the width of the input field.
  - **Default:** 6
  - **Range:** 1–32

- `d` specifies the power of 10 by which to divide the value. If the data contains decimal points, the `d` value is ignored.
  - **Default:** 0
  - **Range:** 0–31

**Details**

The `EUROw.d` informat reads numeric values and removes embedded euro symbols (E), commas, blanks, percent signs, hyphens, and close parentheses from the input data. A decimal point is assumed to be a separator between the whole number and the decimal portion. The `EUROw.d` informat converts an open parenthesis at the beginning of a field to a minus sign.
Comparisons

• The EUROw.d informat is similar to the EUROXw.d informat, but EUROXw.d reverses the roles of the decimal point and the comma. This convention is common in European countries.

• If no commas or periods appear in the input, then the EUROw.d and the EUROXw.d informats are interchangeable.

Example

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

data _null_;  
   input x euro10.;  
   put x=;  
   datalines;  
E1  
E1.23  
1.23  
1,234.56  
run;  
SAS Log:  
x=1  
x=1.23  
x=1.23  
x=1234.56  

<table>
<thead>
<tr>
<th>Values</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| E1     | input x euro10.;  
         | put x;  
         | 1       |
| E1.23  | input x euro10.;  
         | put x;  
         | 1.23    |
| 1.23   | input x euro10.;  
         | put x;  
         | 1.23    |
| 1,234.56 | input x euro10.;  
            | put x;  
            | 1234.56 |

See Also

Formats:

• “EURO Format” on page 97
• “EUROX Format” on page 100
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.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>E1.23</td>
<td>input x eurox10.;</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1.23</td>
<td>input x eurox10.;</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1,234.56</td>
<td>input x eurox10.;</td>
<td>1.23456</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

- “EURO Format” on page 97
- “EUROX Format” on page 100

**Informat:**

- “EURO Informat” on page 416

---

**MINGUO Informat**

Reads dates in Taiwanese format.

**Category:** Date and Time
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 \( yyyyyymmmdd \):

- \( yyyyy \) is an integer that represents the year.

- \( mm \) is an integer from 01 through 12 that represents the month.

- \( dd \) is an integer from 01 through 31 that represents the day of the month.

The Taiwanese calendar uses 1912 as the base year (01/01/01 is January 1, 1912). Dates before 1912 are not valid. Year values do not roll over after 100 years. Instead, they continue to increase.

You can separate the year, month, and day values with any delimiters, such as blanks, slashes, or hyphens, that are permitted by the YYMMDDw. informat. If delimiters are used, place them between all the values. If you omit delimiters, be sure to use a leading zero for days or months that have a value less than 10.

**Example**

The following examples use different dates for input values.

```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>49/01/01</td>
<td>01JAN1960</td>
</tr>
<tr>
<td>891215</td>
<td>15DEC2000</td>
</tr>
</tbody>
</table>
Values | Results  
---|---  
103-01-01 | 01JAN2014

**See Also**

Format:
- “MINGUO Format” on page 102

Informat:
- “YYMMDD Informat” in SAS Viya Formats and Informats: Reference

---

**NENGO Informat**

Reads Japanese date values in the form _eyymmdd_

**Category**: Date and Time

**Syntax**

`NENGO w:`

**Syntax Description**

`w`

specifies the width of the input field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>7–32</td>
</tr>
</tbody>
</table>

**Details**

The general form of a Japanese date is _eyymmdd:

- _e_ is the first letter of the name of the imperial era (Meiji, Taisho, Showa, 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
;
```

<table>
<thead>
<tr>
<th>Values</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

Formats:
- “NENGO Format” on page 104

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**

`NLDATEw`. 

**Syntax Description**

`w` specifies the width of the input field.

Default 20
Example

The following examples use the input February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>15760</td>
</tr>
<tr>
<td>dy='February 24, 2003';</td>
<td></td>
</tr>
<tr>
<td>y=input({'dy,nldate200.});</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>15760</td>
</tr>
<tr>
<td>dy='24. Februar 2003';</td>
<td></td>
</tr>
<tr>
<td>y=input({'dy,nldate16.});</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLDATE Format” on page 107

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

NLDATEW w.

Syntax Description

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

Default 20

Range 16–200
Example

The following examples use the input February 24, 2014.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>19778</td>
</tr>
<tr>
<td>data;</td>
<td></td>
</tr>
<tr>
<td>dy='Monday, February 24, 2014';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldatew200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>19778</td>
</tr>
<tr>
<td>dy='Mo. 24.Februar 2014';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldatew16.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

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**

NLDATMw.

**Syntax Description**

*w*  

specifies the width of the input field.

**Default** 19  
**Range** 19–200

**Example**

The following examples use the input value of February 24, 2003 12:39:43.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></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.

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

Syntax

NLDATMAP\(w\).

Syntax Description

\(w\)

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

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>17–200</td>
</tr>
</tbody>
</table>

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>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>1361709583</td>
</tr>
<tr>
<td>y=input('24.Feb03:12:39:43', nldatm.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>1330171200</td>
</tr>
<tr>
<td>y=input('24.Februar 2003 12.39 Uhr', nldatm.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLDATM Format” on page 128
### 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

```
NLDATMWw
```

### Syntax Description

- **w**
  - Specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.
  - **Default:** 40
  - **Range:** 34–200

### Example

The following examples use the input Mon, Feb 24, 2014 12:39:43 PM.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; data; dy='February 24, 2014 12:39:43 PM'; y=input(dy,nldatmap200.); put y=; run;</td>
<td>1708864783</td>
</tr>
<tr>
<td>options locale=Spanish_Mexico; data; dy='24/02/2003 12:39:43 PM'; y=input(dy,nldatmap200.); put y=; run;</td>
<td>1708864783</td>
</tr>
</tbody>
</table>
Statements | Results
---|---
\[\text{options locale=English\_UnitedStates;}\] | 1708864783
\[\text{data;}\] | 
\[\text{dy='Mon, Feb 24, 2014 12:39:43 PM';}\] | 
\[\text{y=input(dy, nlfmt=200.);}\] | 
\[\text{put y=}\] | 
\[\text{run;}\] | 
\[\text{options locale=German\_Germany;}\] | 1708864783
\[\text{dy='Mo, 24. Feb 2014 12.39 Uhr';}\] | 
\[\text{y=input(dy, nlfmt=16.);}\] | 
\[\text{put y=}\] | 
\[\text{run;}\] | 

**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:** 9
- **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

**Example**

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

\[\text{x=input('($12,345.67$'), nlmniaed32.2);}\]

\[\text{y=input('($12,345.67$'), dollar32.2);}\]
Statements | Results
--- | ---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
- “NLMNLAED Informat” on page 464

NLMNIAUD Informat
Reads the monetary format of the international expression for Australia.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNIAUDw.d

**Syntax Description**

`w`

specifies the width of the output field.

Default: 9

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

**Example**

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

```
x=input('($12,345.67)',nlmniaud32.2);
y=input('($12,345.67)',dollar32.2);
```
## NLMNIBGN Informat

Reads the monetary format of the international expression for Bulgaria.

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

### Syntax

```
NLMNIBGN w.d
```

### Syntax Description

- `w`  
  Specifies the width of the output field.  
  - **Default:** 9  
  - **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

### 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);
```

---

### Statements

<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:**  
- “NLMNLAUD Informat” on page 465
 Statements | Results 
<table>
<thead>
<tr>
<th></th>
<th></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:

- “NLMNLBGN Informat” on page 466

NLMNIBRL Informat

Reads the monetary format of the international expression for Brazil.

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

Syntax

NLMNIBRLw,d

Syntax Description

w

specifies the width of the output field.

Default 9

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

Example

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

```plaintext
x=input('($12,345.67)',nlnibr132.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<table>
<thead>
<tr>
<th></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:**
- “NLMNLBRL Informat” on page 467

---

**NLMNICAD Informat**

Reads 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: 9
  - 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

### Example

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

```plaintext
x=input('($12,345.67)',nlmnicad32.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<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 157

---

**NLMNICHF Informat**

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

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

#### Syntax

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

#### Syntax Description

- **w**  
  Specifies the width of the output field.
  - Default: 9  
  - 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

#### 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);
```
### NLMNICNY Informat

Reads the monetary format of the international expression for China.

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

#### Syntax

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

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 9  
  - 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

#### Example

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

```plaintext
x=input'($12,345.67)',nlmnicny32.2);
y=input'($12,345.67')dollar32.2);
```
### NLMNICZK Informat

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

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

#### Syntax

NLMNICZK\(w.d\)

#### Syntax Description

**\(w\)**  
specifies the width of the output field.  
- **Default:** 9  
- **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

#### Example

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

```plaintext
x=input('($12,345.67)',nlmniczk32.2);
y=input('($12,345.67')',dollar32.2);
```
Statements | Results
-----------+-----------
|-----------+-----------|
| put x=;    | -12345.67 |
| put y=;    | -12345.67 |

See Also

Informat:
- “NLMNCZK Informat” on page 471

NLMNIDKK Informat

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

Category: Numeric
Alignment: Left

Syntax

NLMNIDKKw.d

Syntax Description

w
specifies the width of the output field.
Default 9
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

Example

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

x=input('($12,345.67)',nlmnidkk32.2);
y=input('($12,345.67)',dollar32.2);
Statements | Results
---+---+----+
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:

- “NLMNIDKK Format” on page 161

**NLMNIEEK Informat**

Reads the monetary format of the international expression for Estonia.

Category: Numeric
Alignment: Left

**Syntax**

NLMNIEEK\(_w.d\)

**Syntax Description**

\(w\)

- specifies the width of the output field.
  - Default: 9
  - 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

**Example**

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

```plaintext
x=input('($12,345.67)',nlmnieek32.2);
y=input('($12,345.67)',dollar32.2);
```
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**

- “NLMNLEEK Informat” on page 473

---

**NLMNIEGP Informat**

Reads the monetary format of the international expression for Egypt.

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

### Syntax

```
NLMNIEGPw.d
```

### Syntax Description

- **w**
  - specifies the width of the output field.
  - **Default:** 9
  - **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

### Example

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

```
x=input('($12,345.67)',nlmniegp32.2);
y=input('($12,345.67)',dollar32.2);
```
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
- “NLMNLEGP Informat” on page 474

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 | 9
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

Example

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

```plaintext
x=input'($12,345.67)',nlmnieur32.2);
y=input'($12,345.67)'dollar32.2);
```
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**
- “NLMNIEUR Format” on page 164

### NLMNIGBP Informat

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

**Category:** Numeric

**Alignment:** Left

**Syntax**

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

**Syntax Description**

- \( w \)
  - specifies the width of the output field.
  - **Default:** 9
  - **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

**Example**

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

```plaintext
x=input'($12,345.67)',nlmngbp32.2);
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:

• “NLMNIGBP Format” on page 165

NLMNIHKD Informat

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

Category: Numeric
Alignment: Left

Syntax

NLMNIHKD\textsubscript{\textit{w.d}}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 9
Range 1–32

\textit{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

Example

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

\begin{verbatim}
x=input'($12,345.67)',nlmnihkd32.2);
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:**
- “NLMNIHKD Format” on page 166

---

**NLMNIHRK Informat**

Reads the monetary format of the international expression for Croatia.

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

### Syntax

NLMNIHRK\(w.d\)

### Syntax Description

\(w\)

- specifies the width of the output field.

  - Default: 9
  - 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

### 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 478

NLMNIHUF Informat

Reads the monetary format of the international expression for Hungary.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNIHUF<sub>w,d</sub>

**Syntax Description**

**w**

specifies the width of the output field.

Default 9

Range 1–32

**d**

specifies to divide the number by 10<sup>d</sup>. If the data contains decimal points, the d value is ignored.

Default 0

Range 0–31

**Example**

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

x=input('($12,345.67)',nlnihuf32.2);
y=input('($12,345.67)',dollar32.2);
### NLMNIIDR Informat

Reads the monetary format of the international expression for Indonesia.

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

### Syntax

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

### Syntax Description

- **\(w\)** specifies the width of the output field.
  - Default: 9
  - 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

### Example

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

```plaintext
x=input('\$(12,345.67)',nlniir32.2);
y=input('\$(12,345.67)',dollar32.2);
```
### NLMNIILS Informat

Reads 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: 9  
  - 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

#### Example

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

```plaintext
x=input'($12,345.67)',nlmniils32.2);
y=input'($12,345.67) dollar32.2);
```
### Statements and Results

<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:
- “NLMNIIILS Format” on page 170

### NLMNIIINR Informat

Reads the monetary format of the international expression for India.

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

#### Syntax

NLMNIIINR<sub>w,d</sub>

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 9
  - 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

#### Example

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

```plaintext
x=input('($12,345.67)',nlmninr32.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<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:**
- “NLMNLINR Informat” on page 482

---

### 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: 9
  - 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

### Example

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

```
x=input'($12,345.67)',nlmnijpy32.2);
y=input'($12,345.67)'dollar32.2);
```
### Statements | Results

<table>
<thead>
<tr>
<th></th>
<th></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:**

- “NLMNIJPY Format” on page 172

---

### NLMNIKRW Informat

Reads the monetary format of the international expression for South Korea.

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

#### Syntax

NLMNIKRW<sub>w.d</sub>

#### Syntax Description

**w**

- specifies the width of the output field.
- **Default:** 9  
- **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

#### Example

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

```plaintext
x=input'($12,345.67)',nlmnikrw32.2);
y=input'($12,345.67)‘dollar32.2);
```
See Also

Informat:
- “NLMNLKRW Informat” on page 484

NLMNILTL Informat

Reads the monetary format of the international expression for Lithuania.

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

Syntax

NLMNILTLw.d

Syntax Description

w

specifies the width of the output field.

Default 9

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

Example

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

```plaintext
x=input('($12,345.67)',nlmniltl32.2);
y=input('($12,345.67)',dollar32.2);
```
Statements  |  Results
-----+----1----+
put x=;  |  -12345.67
put y=;  |  -12345.67

See Also

Informat:
- “NLMNLTL Informat” on page 485

NLMNILVL Informat

Reads 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: 9
- 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: \( 0-31 \)

Example

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

\[
x = \text{input}('$(12,345.67)', \text{nlmnilvl}32.2); \\
y = \text{input}('$(12,345.67)', \text{dollar}32.2);
\]
### Statements

<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:

- “NLMNNLLVL Informat” on page 486

---

### 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: 9  
- 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

### Example

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

```plaintext
x=input('($12,345.67)',nlmnimop32.2);
y=input('($12,345.67)',dollar32.2);
```
### NLMNIMXN Informat

Reads 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:** 9  
- **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

### Example

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

```plaintext
x=input('($12,345.67)',nlmnimxn32.2);
y=input('($12,345.67')',dollar32.2);
```
### Statements

<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 488

---

**NLMNIMYR Informat**

Reads the monetary format of the international expression for Malaysia.

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

---

**Syntax**

```
NLMNIMYR w.d
```

**Syntax Description**

- `w` specifies the width of the output field.
  - Default: 9  
  - 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

---

**Example**

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

```plaintext
x=input('($12,345.67)', nlmnimyr32.2);
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 178

---

**NLMNINOK Informat**

Reads 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: 9
  - 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

---

**Example**

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

```plaintext
x=input'($12,345.67)',nlmninok32.2);
y=input'($12,345.67)'dollar32.2);
```
Statements | Results
---+---+----+
| put x=; | -12345.67 |
| put y=; | -12345.67 |

See Also

Format:
- “NLMNINOK Format” on page 179

NLMNINZD Informat
Reads the monetary format of the international expression for New Zealand.

Category: Numeric
Alignment: Left

Syntax

NLMNINZDw.d

Syntax Description

w
specifies the width of the output field.

Default 9
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

Example

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

```
x=input('($12,345.67)',nlmninzd32.2);
y=input'($12,345.67)',dollar32.2);
```
Statements | Results
---|---

---+---

put x=; | -12345.67
put y=; | -12345.67

See Also

Format:

- “NLMNINZD Format” on page 180

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 9
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

Example

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

x=input'($12,345.67)',nlmnipln32.2);
y=input'($12,345.67)'dollar32.2);
See Also

Format:

- “NLMIPLN Format” on page 181

NLMNIRUB Informat

Reads 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 9
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

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>
NLMNISEK Informat

Reads the monetary format of the international expression for Sweden.

Category: Numeric
Alignement: Left

Syntax
NLMNISEKw.d

Syntax Description

w
specifies the width of the output field.
Default 9
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

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnisek32.2); 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:
- “NLMNIRUB Format” on page 182


### Statements

<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:**
- “NLNISEK Format” on page 183

---

### 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:** 9
  - **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

#### Example

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

```plaintext
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>x=</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
See Also

Format:

- “NLMNISGD Format” on page 184

NLMNITHB Informat

Reads 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 9
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

Example

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

x=input('($12,345.67)',nlmnithb32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
- “NLMNLTHB Informat” on page 496

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: 9
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

Example

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

\[\begin{align*}
x &= \text{input}( '(\$12,345.67)' , \text{nlnmtry32.2} ); \\
y &= \text{input}( '(\$12,345.67)' , \text{dollar32.2} );
\end{align*}\]
See Also

Informat:
- “NLMNLTRY Informat” on page 497

NLMNITWD Informat
Reads the monetary format of the international expression for Taiwan.

Category: Numeric
Alignment: Left

Syntax

NLMNITWDw,d

Syntax Description

w
specifies the width of the output field.
Default 9
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

Example

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

```plaintext
x=input'($12,345.67)',nlmmitwd32.2);
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 187

**NLMNIUSD Informat**

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

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNIUSDw.d

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 9
  - 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

**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>
Statements | Results
---+----|
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNIUSD Format” on page 188

**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: 9
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

**Example**

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

\[x=\text{input}'(\$12,345.67)',\text{nlnizar32.2});\]
\[y=\text{input}'(\$12,345.67)'\text{dollar32.2});\]

| Statements | Results |
|---+----|
| - - - - - - - - - - - - - - - - - - - - |
| - - - - - - - - - - - - - - - - - - - - |
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

**See Also**

**Format:**
- “NLMNIZAR Format” on page 189

---

**NLMNLAED Informat**

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

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNLAED\(w,d\)

**Syntax Description**

\(w\)

specifies the width of the output field.

- **Default:** 9
- **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

**Example**

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

```plaintext
x=input('($12,345.67)',nlmnlaed32.2);
y=input('($12,345.67)',dollar32.2);
```

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


<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 427

**NLMNLAUD Informat**

Reads the monetary format of the local expression for Australia.

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

**Syntax**

NLMNLAUD\_\_w.d\__

**Syntax Description**

\_w\__

specifies the width of the output field.

- **Default:** 9
- **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

**Example**

In the following example, the LOCALE= system option is set to English United States.

x=input('($12,345.67)',nlmnlaud32.2);
y=input('($12,345.67)',dollar32.2);

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

Format:
• “NLMNLAUD Format” on page 191

### NLMNLBGN Informat

Reads the monetary format of the local expression for Bulgaria.

**Category:** Numeric

**Alignment:** Left

#### Syntax

```plaintext
NLMNLBGN w.d
```

#### Syntax Description

- **w**
  - specifies the width of the output field.
  - **Default:** 9
  - **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

#### Example

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

```plaintext
x=input(-12345.67,nlmnlbgn32.2);
y=input(-12345.67,dollar32.2);
```

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

---+----1----+

---+----1----+

---+----1----+
NLMNLBRL Informat

Reads the monetary format of the local expression for Brazil.

**Category:** Numeric

**Alignment:** Left

### Syntax

NLMNLBRL<sub>w.d</sub>

### Syntax Description

- **w**  
  Specifies the width of the output field.
  
  **Default:** 9  
  **Range:** 1–32

- **d**  
  Specifies to divide the number by 10<sup>d</sup>. If the data contains decimal points, the d value is ignored.
  
  **Default:** 0  
  **Range:** 0–31

### Example

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

```plaintext
x=input('($12,345.67)',nlmnlbrl32.2);  
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Informat:**

- “NLMNIBGN Informat” on page 429
<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:**
- “NLMNIBRL Informat” on page 430

### NLMNLCAD Informat

Reads 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:** 9
- **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

**Example**

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

```plaintext
x=input('($12,345.67)',nlmnlcad32.2);
y=input('($12,345.67)',dollar32.2);
```

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

---

---
### NLMNLCHF Informat

Reads 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: 9
  - 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

#### Example

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

```plaintext
x=input('($12,345.67)',nlmnlchf32.2);
y=input('($12,345.67)',dollar32.2);
```

---

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

### See Also

**Format:**
- “NLMNLCAD Format” on page 194
<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:**
- “NLMNLCHF Format” on page 195

---

**NLMNLNCY Informat**

Reads the monetary format of the local expression for China.

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

**Syntax**

\[ \text{NLMNLNCY}_{w,d} \]

**Syntax Description**

- \( w \) specifies the width of the output field.
  - **Default:** 9
  - **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

**Example**

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

\[
\begin{align*}
x &= \text{input}('($12,345.67')$, nlmnlcny32.2); \\
y &= \text{input}('($12,345.67')$, dollar32.2);
\end{align*}
\]
## Statements

<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

**Format:**

- “NLMNLCNY Format” on page 196

---

## NLMNLCZK Informat

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

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

### Syntax

`NLMNLCZKw,d`

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **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

### Example

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

```plaintext
x=input('($12,345.67)',nlnlczk32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
**Statements** | **Results**
--- | ---
put x=; | -12345.67
put y=; | -12345.67

**See Also**

**Informat:**
- “NLMNICZK Informat” on page 434

---

**NLMNLDKK Informat**

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

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

**Syntax**

NLMNLDKK\(w.d\)

**Syntax Description**

\(w\)

- Specifies the width of the output field.
  - Default: 9
  - 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

**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);
```

---

**Statements** | **Results**
--- | ---

--- +---1---+

NLMNLEEK Informat

Reads the monetary format of the local expression for Estonia.

Category: Numeric
Alignment: Left

Syntax

NLMNLEEKw.d

Syntax Description

w

specifies the width of the output field.

Default  9
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

Example

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

x=input('($12,345.67)',nlmnleek32.2);
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:

- “NLMNLDDK Format” on page 198
**Statements** | **Results**
---|---
`put x=;` | -12345.67
`put y=;` | -12345.67

**See Also**

**Informat:**
- “NLMNIEEK Informat” on page 436

---

**NLMNLEGIP Informat**

Reads the monetary format of the local expression for Egypt.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

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

**Syntax Description**

\[ w \]

specifies the width of the output field.

- **Default:** 9
- **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

---

**Example**

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

```plaintext
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></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
- “NLMNIEGP Informat” on page 437

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_{\textit{w}}.\textit{d}

**Syntax Description**

\textit{w}

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\textit{d}

specifies to divide the number by $10^{\textit{d}}$. If the data contains decimal points, the \textit{d} value is ignored.

- **Default:** 0
- **Range:** 0–31

**Example**

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

```plaintext
x=input$('($12,345.67)',nlmnleur32.2);
y=input$('($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 201

---

**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:** 9
- **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

**Example**

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

\[
\begin{align*}
  x &= \text{input}(\text{'$(12,345.67)'}, \text{nlmnlgbp32.2}); \\
  y &= \text{input}(\text{'$(12,345.67)'}, \text{dollar32.2});
\end{align*}
\]
### Statements

<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:
- “NLMNGBP Format” on page 202

---

**NLMNLHKD Informat**

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

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

#### Syntax

```
NLMNLHKD w.d
```

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 9
  - 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

#### Example

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

```plaintext
x=input('($12,345.67)',nlmnlhkd32.2);
y=input('($12,345.67)',dollar32.2);
```
Statements                  Results

-------+-------+-------

<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:

- “NLMNLHKD Format” on page 203

NLMNLHRK Informat

Reads the monetary format of the local expression for Croatia.

Category: Numeric
Alignment: Left

Syntax

NLMNLHRK\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

- Default: 9
- 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

Example

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

\(x=\)input('($12,345.67)',nmlhrk32.2);
\(y=\)input('($12,345.67')',dollar32.2);
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “NLMNIHRK Informat” on page 441

### NLMNLHUF Informat

Reads the monetary format of the local expression for Hungary.

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

#### Syntax

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

#### Syntax Description

- \( w \)
  - Specifies the width of the output field.  
  - **Default:** 9  
  - **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

#### Example

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

\[
x=\text{input}(\'(\$12,345.67)\',\text{nlmnlhuf32.2});
y=\text{input}(\'(\$12,345.67)\',\text{dollar32.2});
\]
<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:

- “NLMNIHUF Informat” on page 442

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: 9
  - 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

Example

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

```plaintext
x=input('($12,345.67)',nlmnlidr32.2);
y=input('($12,345.67)',dollar32.2);
```
### NLMNLILS Informat

Reads the monetary format of the local expression for Israel.

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

#### Syntax

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

#### Syntax Description

- **\( w \)**: specifies the width of the output field.  
  - Default: 9  
  - 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

#### Example

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

```plaintext
x=input('($12,345.67)',nlmnlils32.2);
y=input('($12,345.67)',dollar32.2);
```

---

### See Also

**Informat:**  
- “NLMNIIDR Informat” on page 443

---

<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>
Statements | Results
---|---
---+----1----+
**put** x=; | -12345.67
**put** y=; | -12345.67

See Also

**Format:**

- “NLMN LILS Format” on page 207

---

**NLMN LINR Informat**

Reads the monetary format of the local expression for India.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMN LINR \( w.d \)

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 9
- **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

---

**Example**

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

\[
x=\text{input}('$(12,345.67)',\text{nlmnlir}32.2);
y=\text{input}('$(12,345.67)',\text{dollar}32.2);
\]
Statements | Results
---|---
| -12345.67 | -12345.67

See Also

Informat:
- “NLMNIINR Informat” on page 445

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 9
  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

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);
```
### 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: 9  
  - 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

### 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);
```

---

**See Also**

**Format:**
- “NLMNLJPY Format” on page 209
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>-12345.67</td>
</tr>
<tr>
<td></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:
- “NLMIKRW Informat” on page 447

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 9
- 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

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.
\[
x=\text{input}('($12,345.67)',\text{nlmnlltl}132.2);
y=\text{input}('($12,345.67)',\text{dollar}32.2);
\]
**NLMNLLVL Informat**

Reads the monetary format of the local expression for Latvia.

**Category:** Numeric

**Alignment:** Left

### Syntax

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

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 9
  - 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

### 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);
```

**See Also**

Informat:
- “NLMNILTL Informat” on page 448

---

<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:

• “NLMNVLVL Informat” on page 449

NLMNLMOP Informat

Reads the monetary format of the local expression for Macau.

- Category: Numeric
- Alignment: Left

Syntax

NLMNLMOP_w.d

Syntax Description

w

specifies the width of the output field.

- Default: 9
- 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

Example

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

```plaintext
x=input('($12,345.67)',nlmnlmop32.2);
y=input('($12,345.67)',dollar32.2);
```
See Also

Informat:

- “NLMNIMOP Informat” on page 450

NLMNLMXN Informat

Reads the monetary format of the local expression for Mexico.

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

Syntax

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

Syntax Description

- **\( w \)** specifies the width of the output field.
  
  Default: 9
  
  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

Example

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

\[
\begin{align*}
x &= \text{input}(\'\$(12,345.67)', \text{nlmnlnxn32.2}); \\
y &= \text{input}(\'\$(12,345.67)', \text{dollar32.2});
\end{align*}
\]
### Statements

<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

**Informat:**

- “NLMNIMXN Informat” on page 451

---

### NLMNLMYR Informat

Reads the monetary format of the local expression for Malaysia.

#### Category:

Numeric

#### Alignment:

Left

### Syntax

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

#### Syntax Description

#### \( w \)

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

- **Default**: 9
- **Range**: 1–32

#### \( d \)

\( d \) specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

- **Default**: 0
- **Range**: 0–31

### 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);
```
NLMNLNOK Informat
Reads the monetary format of the local expression for Norway.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNLNOK\(_w.d\)

**Syntax Description**

\(w\)

specifies the width of the output field.

- **Default:** 9
- **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

**Example**

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

```plaintext
x=input('($12,345.67)',nlmlnok32.2);
y=input('($12,345.67)',dollar32.2);
```

---

### Statements

<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:**

- “NLMNLMYR Format” on page 215
Statements | Results
---+----1----+
| put x=; | -12345.67 |
| put y=; | -12345.67 |

See Also

Format:

- “NLMNLNOK Format” on page 216

NLMNLNZD Informat

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

Category: Numeric
Alignment: Left

Syntax

NLMNLNZD \( w.d \)

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
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

Example

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

\( x=\)input('($12,345.67)',nlmnlnzd32.2);
\( y=\)input('($12,345.67)',dollar32.2);
Statements | Results  
---+----1----+  
put x=; | -12345.67  
put y=; | -12345.67

See Also

Format:
- “NLMNLNZD Format” on page 217

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 9
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

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);

Statements | Results  
---+----1----+  
put x=; | -12345.67  
put y=; | -12345.67
See Also

Format:

• “NLMNLPLN Format” on page 218

NLMNLPRUB Informat

Reads the monetary format of the local expression for Russia.

Category: Numeric
Alignment: Left

Syntax

NLMNLPRUB\(_w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default 9
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

Example

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

\[
x = \text{input}(\text{'($12,345.67')', nlmnlrub32.2});\\
y = \text{input}(\text{'($12,345.67')', dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=)</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put (y=)</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNLSEK Format” on page 219

NLMNLSEK Informat
Reads the monetary format of the local expression for Sweden.

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

**Syntax**

NLMNLSEK<sub><var>w</var>,<var>d</var></sub>

**Syntax Description**

<var>w</var>
- Specifies the width of the output field.
  - Default: 9
  - Range: 1–32

<var>d</var>
- Specifies to divide the number by 10<sup><var>d</var></sup>. If the data contains decimal points, the <var>d</var> value is ignored.
  - Default: 0
  - Range: 0–31

**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>-----</td>
<td>---</td>
</tr>
<tr>
<td>x=</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
NLMNLSGD Informat

Reads the monetary format of the local expression for Singapore.

**Category:** Numeric

**Alignment:** Left

### Syntax

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

#### Syntax Description

- **w** specifies the width of the output field.
  - Default: 9
  - 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

### Example

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

```plaintext
x=input('($12,345.67)',nlmnlsgd32.2);
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:**

- “NLMNLSEK Format” on page 220
See Also

Format:

- “NLMNLSGD Format” on page 221

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 9
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

Example

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

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 459

### NLMNLTRY Informat

Reads 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:** 9  
  **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

**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 460

NLMNLTWD Informat

Reads the monetary format of the local expression for Taiwan.

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

Syntax

NLMNLTWD w.d

Syntax Description

w

specifies the width of the output field.

Default: 9

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

Example

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

x=input('($12,345.67)',nlmnltwd32.2);
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>
**Statements** | **Results**
---|---
put x=; | -12345.67
put y=; | -12345.67

**See Also**

**Format:**
- “NLMLNLTWD Format” on page 224

---

**NLMLNUSD Informat**

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

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

**Syntax**

NLMNLUSD\_w\_d

**Syntax Description**

\_w\_

- specifies the width of the output field.
  - **Default:** 9
  - **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

**Example**

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

\```sas\n\x=\text{input}('($12,345.67)',nlmlnusd32.2); \y=\text{input}('($12,345.67)',dollar32.2);\```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
**Statements** | **Results**
---|---
put x=; | -12345.67
put y=; | -12345.67

### See Also

**Format:**
- “NLMNLUSD Format” on page 225

---

**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: 9
  - 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

**Example**

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

```plaintext
x=input('($12,345.67)',nlmnlzar32.2);
y=input'($12,345.67)',dollar32.2);
```

**Statements** | **Results**
---|---
--- | ----+----
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNLZAR Format” on page 226

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

NLMNY_{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 NLMNY_{w,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 NLMNY_{w,d} informat performs processing that is the opposite of the NLMNY1_{w,d} informat.

The NLMNY_{w,d} informat is similar to the DOLLAR_{w,d} informat except that the NLMNY_{w,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.

<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></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 227
- “NLMNYI Format” on page 228

Informat:
- “NLMNYI Informat” on page 502

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**

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></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 227
• “NLMNYI Format” on page 228

Informat:

• “NLMNY Informat” on page 501
NLNUM Informat

Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

NLNUM\(w.d\)

**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 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 \(-123456.78\) as the input value.
Statements | Results
---+----1----+

```latex
options locale=English_UnitedStates;
x=input('{-1,234,356.78}',nlnum32.2);
put x=;
```

### See Also

**Formats:**
- “NLNUM Format” on page 229
- “NLMNYI Format” on page 228

**Informat:**
- “NLNUMI Informat” on page 505

---

**NLNUMI Informat**

Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.

**Category:** Numeric

---

**Syntax**

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

**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 \( \text{NLNUMI}_{w.d} \) 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 NLNUMIw.d informat performs processing that is opposite of the NLNUMw.d informat.

Example
The following example uses –1,234,356.78 as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>-1234356.78</td>
</tr>
<tr>
<td>x=input('-1,234,356.78', nlnumi32.2);</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also
Formats:
- “NLNUM Format” on page 229
- “NLNUMI Format” on page 231
- “NLNUM Informat” on page 504

NLPCT Informat
Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.

| Category: | Numeric |

Syntax
NLPCTw.d

Syntax Description

\(w\)

- specifies the width of the input field.

Default: 6

Range: 1–32
\( d \)
specifies whether to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.

**Default** 0

**Range** 0–31

### Details
The NLPCTw.\(d\) 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 NLPCTw.\(d\) informat performs processing that is opposite of the NLPCTIw.\(d\) informat. The NLPCTw.\(d\) informat is similar to the PERCENTw.\(d\) informat except that the NLPCTw.\(d\) 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=;
rn;

### 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(&quot;-12,345.67%&quot;,nlpc32.2);</td>
<td>-123.4567</td>
</tr>
<tr>
<td>y=input(&quot;{(12,345.67%)&quot;,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 232
- “NLPCTI Format” on page 233
- “NLPCTI Informat” on page 508
NLPCTI Informat

Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

NLPCTI\(w.d\)

**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></td>
<td>-1.234567</td>
</tr>
</tbody>
</table>
Statements | Results
---|---
options LOCALE=English_UnitedStates; | -123.4567
x=input(''-12,345.67%',nlpct32.2); | -123.4567
y=input(''{12,345.67%}' ,percent32.2)); | 
put x=; | 
put y=; | 

See Also

Formats:
- “NLPCT Format” on page 232
- “NLPCTI Format” on page 233

Informat:
- “NLPCT Informat” on page 506

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\textsubscript{w}.

**Syntax Description**

\texttt{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></td>
<td>---</td>
</tr>
</tbody>
</table>


Statements | Results
---|---
```plaintext
options locale=English_UnitedStates;
y=input('04:24:43 PM',nltimap11.);
put y time.;
```
| 16:24:43 |
---|---

```plaintext
options locale=German_Germany;
y=input('16.24 Uhr',nltimap11.);
put y time.;
```
| 16:24:00 |
---|---

### 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.

<table>
<thead>
<tr>
<th>Category</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alias:</strong></td>
<td>NLTMAP</td>
</tr>
</tbody>
</table>

**Syntax**

NLTIME

**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.

```plaintext
options locale=English_UnitedStates;
y=input('16:24:43',nltime.);
put y time.;
```

<table>
<thead>
<tr>
<th>16:24:43</th>
</tr>
</thead>
</table>

### Syntax

NLTIME

**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.

```plaintext
options locale=English_UnitedStates;
y=input('16:24:43',nltime.);
put y time.;
```

<table>
<thead>
<tr>
<th>16:24:43</th>
</tr>
</thead>
</table>
Statements

```plaintext
options locale=German_Germany;
y=input('16.24 Uhr',nltime.);
put y time.;
```

Results

```
16:24:00
```

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**

```plaintext
$UCS2B
```

**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** 2–32000

**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></td>
<td>----+----1----+</td>
</tr>
<tr>
<td></td>
<td>------+-+------</td>
</tr>
<tr>
<td></td>
<td>16:24:00</td>
</tr>
</tbody>
</table>
$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** 1–32000

### 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>
<tbody>
<tr>
<td>ucs2str=input ('פע', $ucs2be4.); put ucs2str=$hex8.;</td>
<td>ucs2str=00205927</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2B Format” on page 243
- “$UCS2BE Format” on page 245

Informat:
- “$UCS2B Informat” on page 511

$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.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>UTF-8 is the only SAS session encoding supported by SAS Viya.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2–32000</td>
</tr>
</tbody>
</table>

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,$ucs2l.);</td>
<td>x=E5A4A7</td>
</tr>
<tr>
<td>put x=$hex6.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2B Format” on page 243
- “$UCS2L Format” on page 246
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 265

Informats:
- “$UCS2B Informat” on page 511
- “$UCS2X Informat” on page 515
- “$UTF8X Informat” on page 530

$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: 1–32000

Comparisons
The $UCS2LE_w$ informat performs processing that is opposite of the $UCS2L_w$ informat.

Example
This example uses UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input (' arrayWithUTF8Encoding', $ucs2le4.);</td>
<td>ucs2str=2759</td>
</tr>
<tr>
<td>put ucs2str=$hex4;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2L Format” on page 246
- “$UCS2LE Format” on page 247

Informat:
- “$UCS2L Informat” on page 513

$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
$UCS2X_w$

Syntax Description

w specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.
Range  2–32000

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 246
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 265

Informats:
- “$UCS2B Informat” on page 511
- “$UCS2L Informat” on page 513
- “$UTF8X Informat” on page 530

$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 1-32000

Comparisons

The \$UCS2XE\( w \) informat performs processing that is opposite of the \$UCS2X\( w \) 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 ('-cancel ', $ucs2xe4.);</td>
<td>ucs2str=20002759</td>
</tr>
<tr>
<td>put ucs2str=$hex8;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “\$UCS2X Format” on page 248
- “\$UCS2XE Format” on page 250

Informat:
- “\$UCS2X Informat” on page 515

\$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

\$UCS4B\( w \).
**Syntax Description**

\( w \)

specifies the width of the input 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>4</td>
<td>4–32000</td>
</tr>
</tbody>
</table>

**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>\begin{verbatim} ucs4str=input('✘', $ucs4be8.); put ucs4str= $hex16.; \end{verbatim}</td>
<td>\begin{verbatim} ucs4str=0000002000005927 \end{verbatim}</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “\( $UCS4B \) Format” on page 251

**Informats:**

- “\( $UCS4L \) Informat” on page 518
- “\( $UCS4X \) Informat” on page 519

---

**\( $UCS4L \) 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 4
Range 4–32000

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>2E00000063000006F0000000D000000</td>
</tr>
<tr>
<td>put z $hex32.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “$UCS4L Format” on page 253

Informats:

- “$UCS4B Informat” on page 517
- “$UCS4X Informat” on page 519

$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 Description

$w$

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

- **Default**: 4
- **Range**: 4–32000

### Comparisons

The $\text{UCS4X}w$ informat performs processing that is the opposite of the $\text{UCS4XE}w$ informat. Use the $\text{UCS4X}w$ informat when you are processing data within the same operating environment. Use the $\text{UCS4B}w$ and $\text{UCS4L}w$ 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></td>
<td>-----+-----1----+</td>
</tr>
<tr>
<td>ucs4=put(‘91e5’x, $ucs4x.);</td>
<td>ucs4=FDFF0000</td>
</tr>
<tr>
<td>utf8=input(ucs4, $ucs4x.);</td>
<td>utf8=EFBFBD20</td>
</tr>
<tr>
<td>put ucs4=$\text{hex8}$. utf8=$\text{hex8}$.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Formats:**

- “$\text{UCS2X Format}$” on page 248
- “$\text{UCS2B Format}$” on page 243
- “$\text{UCS2L Format}$” on page 246
- “$\text{UCS4X Format}$” on page 256
- “$\text{UTF8X Format}$” on page 265

**Informats:**

- “$\text{UCS2B Informat}$” on page 511
- “$\text{UCS2L Informat}$” on page 513
- “$\text{UTF8X Informat}$” on page 530
$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:** 1–32000

### 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('ü', $ucs4xe8.); put ucs4str=$hex16;</td>
<td>ucs4str=2000000027590000</td>
</tr>
</tbody>
</table>

### See Also

**Formats:**

- “$UCS4X Format” on page 256  
- “$UCS4XE Format” on page 257

**Informat:**

- “$UCS4X Informat” on page 519
$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**

$UESC_{w}.

**Syntax Description**

\(w\)

specifies the width of the output field.

- **Default:** 8
- **Range:** 1–32000

**Details**

If the characters are not available on all operating environments (for example, 0–9, a–z, A–Z) they must be represented in UESC representation. The $UESC_{w}$. informat can be nested.

**Comparisons**

The $UESC_{w}$. informat performs processing that is the opposite of the $UESCE_{w}$. informat.

**Example**

These examples use the 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;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>
$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**

\[ \text{SUESCE}_w. \]

**Syntax Description**

\( w \)

- specifies the width of the input field.

  - Default: 8
  - Range: 1–32000

**Details**

The \( \text{SUESCE}_w. \) informat can be nested.

**Comparisons**

The \( \text{SUESCE}_w. \) informat performs processing that is opposite of the \( \text{SUEC}_w. \) 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=\text{input(' \text{&quot;&quot;, } \text{uescel0.}.)} )</td>
<td>\text{\textbackslash u5927}</td>
</tr>
<tr>
<td>( y=\text{input('Yu5927',}\text{uescel0.};) )</td>
<td></td>
</tr>
</tbody>
</table>

See Also

- “$UESC Format” on page 258
- “$UESCE Format” on page 259

Informat:

- “$UESCE Informat” on page 523
See Also

Formats:

- “$UESC Format” on page 258
- “$UESCE Format” on page 259

Informat:

- “$UESC Informat” on page 522

$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.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1–32000</td>
</tr>
</tbody>
</table>

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 $UNCRe w. informat.

Example

These examples use UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>------</td>
</tr>
</tbody>
</table>
### Statements Result

<table>
<thead>
<tr>
<th>x = input ('&amp;#22823;', $unacr10.);</th>
<th>€</th>
</tr>
</thead>
<tbody>
<tr>
<td>y = input ('abc', $unacr10.);</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 525

---

### $UNCRE Informat

Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.

<table>
<thead>
<tr>
<th>Category</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>UTF-8 is the only SAS session encoding supported by SAS Viya.</td>
</tr>
</tbody>
</table>

### Syntax

\[
\text{$UNCRE}\_w. 
\]

### Syntax Description

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

<table>
<thead>
<tr>
<th>Default</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>1–32000</td>
</tr>
</tbody>
</table>

### Details

The output string converts to plain characters and NCR. Any national characters convert to NCR.

### Comparisons

The $UNCRE\_w. informat performs processing that is the opposite of the $UNCR\_w. informat.
Example

These examples use UTF-8 encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x=input ('abc', $uncre12.);</code></td>
<td><code>$$22823;abc</code></td>
</tr>
<tr>
<td><code>put x;</code></td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UNCR Format” on page 260
- “$UNCRE Format” on page 261

Informat:
- “$UNCR Informat” on page 524

$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 1–32000

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;',$uparen10.);</td>
<td>$</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:

- “$SUPAREN Format” on page 262
- “$SUPARENE Format” on page 264

Informats:

- “$SUPARENE Informat” on page 527
- “$SUPARENP Informat” on page 528

$SUPARENE 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

$SUPARENEw$.

Syntax Description

$w$

specifies the width of the input field.
Comparisons

The $UPARENE_{w}$ 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>v=input('a',$uparene10.);</td>
<td>&lt;u0061&gt;</td>
</tr>
<tr>
<td>w=input('b',$uparenie10.);</td>
<td>&lt;u0062&gt;</td>
</tr>
<tr>
<td>x=input('c',$uparene10.);</td>
<td>&lt;u0063&gt;</td>
</tr>
<tr>
<td>y=input('3',$uparene10.);</td>
<td>&lt;u0033&gt;</td>
</tr>
<tr>
<td>z=input('.stub', $uparene10.);</td>
<td>&lt;u5927&gt;</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$SUPAREN Format” on page 262
- “$SUPARENE Format” on page 264

Informats:
- “$SUPAREN Informat” on page 526
- “$SUPARENP Informat” on page 528

$SUPARENP 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.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>UTF-8 is the only SAS session encoding supported by SAS Viya.</td>
</tr>
</tbody>
</table>
Syntax

SUPARENP

Syntax Description

$w$

specifies the width of the input field.

Default 8

Range 1–32000

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;',$uparenp10.);</td>
<td>&lt;u5927&gt;</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

- “SUPAREN Format” on page 262
- “SUPARENE Format” on page 264

Informats:

- “SUPAREN Informat” on page 526
- “SUPARENE Informat” on page 527
$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**

$UTF8X

**Syntax Description**

$w

w specifies the width of the input field.

- **Default:** 8
- **Range:** 1–32000

**Comparisons**

This example uses the Japanese UTF-8 encoding, which is supported under the Linux operating environment.

```
x=input ('e5a4a7'x, $utf8x3.);
put x;
```

**See Also**

- “$UCS2B Format” on page 243
- “$UCS2L Format” on page 246
- “$UCS2X Format” on page 248
- “$UTF8X Format” on page 265

**Formats:**

- “$UCS2B Format” on page 511
- “$UCS2L Format” on page 513
- “$UCS2X Format” on page 515
Part 8

Macro Functions for NLS

Chapter 16

Dictionary of Macro Functions for NLS ............................ 533
Chapter 16
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. 534)</td>
<td>Compresses multiple blanks and removes leading and trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%KINDEX Macro Function (p. 534)</td>
<td>Returns the position of the first character of a string.</td>
</tr>
<tr>
<td></td>
<td>%KLEFT and %QKLEFT Macro Functions (p. 535)</td>
<td>Left-aligns an argument by removing leading blanks.</td>
</tr>
<tr>
<td></td>
<td>%KLENGTH Macro Function (p. 535)</td>
<td>Returns the length of a string.</td>
</tr>
<tr>
<td></td>
<td>%KSCAN and %QKSCAN Functions (p. 536)</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. 538)</td>
<td>Produce a substring of 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>%KUPCASE and %QKUPCASE Macro Functions (p. 540)</td>
<td></td>
<td>Convert values to uppercase.</td>
</tr>
</tbody>
</table>

## 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:
    %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:

\[
& \ $ \ ^ \ ( \ ) \ + \ - \ * \ / \ < \ > \ = \ \neg \ ^ \ ^ \ ^ \ ; \ , \ # \ \text{blank}
\]

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:
  
  \[
  \text{blank} \ ! \ $ \ & \ ( \ ) \ + \ , \ . \ / \ ; \ < \ ^ \ ^ \ ^ \ ^ \ ^ \ ^ \ ^ \ \text{~}
  \]

  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:
  
  \[
  \text{blank} \ ! \ $ \ & \ ( \ ) \ + \ , \ . \ / \ ; \ < \ \text{¬} \ \text{|} \ \text{¢}
  \]

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:

```n
%macro a;
  aaaaaa
%mend a;
%macro b;
  bbbbbbb
%mend b;
%macro c;
  cccccccc
%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: cccccc
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 %EVAL function or %IF condition where a numeric operand is required.
Message 2 is: A character operand was found in the %EVAL function or %IF condition where a numeric operand is required.
Message 3 is: A character operand was found in the %EVAL function or %IF condition where a numeric operand is required.
Message 4 is: A character operand was found in the %EVAL function or %IF condition where a numeric operand is required.
Message 5 is: A character operand was found in the %EVAL function or %IF condition where a numeric operand is required.

Example 3: Comparing the Actions of %KSUBSTR and %QKSUBSTR
%KSUBSTR produces a resolved result because it does not mask special characters and mnemonic operators in the C language before processing it:

```sas
%let a=one;
```
%let b=two;
%let c=%nrstr(&a &b);
%put C: &c;
%put With KSUBSTR: %ksubstr(&c,1,2);
%put With QKSUBSTR: %qKsubstr(&c,1,2);

When these statements execute, these lines are written to the SAS log:

C: &a &b
With KSUBSTR: one
With QKSUBSTR: &a

---

**%KUPCASE and %QKUPCASE Macro Functions**

Convert values to uppercase.

**Category:** DBCS

**Type:** NLS macro function

**Syntax**

%KUPCASE (character string | text expression)

%QKUPCASE (character string | text expression)

**Details**

The %KUPCASE and %QKUPCASE functions convert lowercase characters in the argument to uppercase. %KUPCASE does not mask special characters or mnemonic operators in its results.

If the argument contains a special character or mnemonic operator, listed here, use %QKUPCASE. %QKUPCASE masks the following special characters and mnemonic operators in its results:

& % ' " ( ) + * / < > = ¬ ^ ~ ; , # blank
AND OR NOT EQ 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.

```sas
%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:

```sas
%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
System Option Entries by Category

The language control category of SAS system options are affected by NLS. The following table provides brief descriptions of the SAS system options. For more detailed descriptions, see the dictionary entry for each SAS system option:

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment Control: Language</td>
<td>DFLANG= System Option (p. 547)</td>
<td>Specifies the language for international date informats and formats.</td>
</tr>
<tr>
<td></td>
<td>ENCODING System Option (p. 549)</td>
<td>Specifies the default character-set encoding for the SAS session.</td>
</tr>
<tr>
<td></td>
<td>LOCALE System Option (p. 550)</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. 551)</td>
<td>Specifies the source database for the locale information.</td>
</tr>
<tr>
<td></td>
<td>LOGLANGENG System Option (p. 553)</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. 555)</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. 557)</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. 560)</td>
<td>Specifies the user local time zone.</td>
</tr>
<tr>
<td>Files: External Files</td>
<td>BOMFILE System Option (p. 546)</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. 558)</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. 561)</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. 564)</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. 554)</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. 559)</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.

```sas
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.

```sas
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
Sas.exe -locale French
Proc print data=sashelp.class ; run ;
```

```sas
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 599.

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 599

System Options:
• “ENCODING System Option” on page 549
• “DFLANG= System Option” on page 547

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 557
- “LSWLANG System Option” on page 554
- “LOGLANGENG System Option” on page 553
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 554
• “LOGLANGCHG System Option” on page 552
• “ODSLANGCHG System Option” on page 557

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=LOCALSE | language

Required Arguments

LOCALSE
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>
<tr>
<td>Code</td>
<td>Language</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>JA</td>
<td>Japanese</td>
</tr>
<tr>
<td>KO</td>
<td>Korean</td>
</tr>
<tr>
<td>PL</td>
<td>Polish</td>
</tr>
<tr>
<td>RU</td>
<td>Russian</td>
</tr>
</tbody>
</table>

See Also

- “ODSLANGCHG System Option” on page 557
- “LOGLANGCHG System Option” on page 552
- “LOGLANGENG System Option” on page 553

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 88

ODSLANGCHG System Option
Determines whether the language of the text of the ODS output can be changed

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control

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 appear 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 appear 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 553
- “LOGLANGCHG System Option” on page 552
- “LSWLANG System Option” on page 554

---

**RSASIOTRANSERROR System Option**

Displays a transcoding error when invalid data is read from a remote application.

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Files: SAS Files

**PROC OPTIONS GROUP=** SASFILES

**Default:** RSASIOTRANSERROR

**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 Utility Procedures Guide for more information.

**Example**

This example demonstrates the functionality of SORTSEQ with PROC SORT and PROC SQL:

```
options sortseq=linguistic;
proc sort data=sashelp.class out=foo1;
  by name;
run;
proc sql;
  create table foo2 as select * from sashelp.class order by name;
quit;
run;
```
SAS provides ICU collation when the linguistic option (SORTSEQ=LINGUISTIC) is specified 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=

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
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 1, “Time Zone IDs and Time Zone Names,” on page 625.

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 1, “Time Zone IDs and Time Zone Names,” on page 625.

Details

You set the TIMEZONE= option to a time zone ID or a time zone name in order for SAS to use a particular time zone. The time zone setting affects the following SAS components:

- times that are recorded by events and logs
- time of data set creation or modification
- DATE( ) function
- DATETIME( ) function
- TIME( ) function
- TODAY( ) function
- time zone functions TZONEOFF( ), TZONEID( ), TZONENAME( ), TZONES2U( ), and TZONEU2S( ),
- time zone formats B8601DXw., E8601DXw., B8601LXw., E8601LXw., B8601TXw., E8601TXw., NLDATMZw., NLDATMTZw., and NLDATMWZw.

You set a time zone by specifying a time zone ID or a time zone name. A time zone ID is a region and an area separated by a forward slash (/). For example, America/New_York and Asia/Osaka are time zone IDs.

A time zone name is a three- or four-character name for a time zone. For example, EST is Eastern Time and JST is Japan Time. SAS determines the time by using time zone rules, including daylight saving time rules, before using a time value.

Some time zones names are valid for different locales. For example, CST is Central 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

VALIDMEMNAME= System Option

Specifies the rules for naming SAS data sets, SAS data views, and item stores.

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 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"n;

data ‘Años de empleo’n.;

**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 564
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 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.

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 VALIDVARNAME=ANY system option is to enable compatibility with other DBMS variable (column) naming conventions, such as allowing embedded blanks and national characters.

See Also

System Options:

• “VALIDMEMNAME= System Option” on page 561
Part 10

Options for Commands, Statements, and Procedures for NLS

Chapter 18

Dictionary of Command, Statement, and Procedure Option for NLS

569
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. 577)</td>
<td>Specifies attributes for character variables that are needed in order to transcode a SAS file.</td>
</tr>
<tr>
<td></td>
<td>ENCODING= Option (p. 582)</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. 585)</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></td>
<td>ODSCHARSET= Option (p. 587)</td>
<td>Specifies the character set to be generated in the META declaration for the output.</td>
</tr>
<tr>
<td></td>
<td>ODDSTRANTAB= Option (p. 588)</td>
<td>Specifies the translation table to use when transcoding an XML document for an output file.</td>
</tr>
<tr>
<td></td>
<td>XMLENCODING= Option (p. 592)</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. 589)</td>
<td>Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and length with one or more variables) that indicates whether character variables are to be transcoded.</td>
</tr>
<tr>
<td>ODS: Third-Party Formatted</td>
<td>CHARSET= Option (p. 570)</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. 591)</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

```html
<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 18.1 on page 572.

**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 18.1 on page 572.

**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 18.1 Alphanumeric Characters Sorted for Each Language**

<table>
<thead>
<tr>
<th>Language</th>
<th>Character Sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Danish</td>
<td>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz</td>
</tr>
<tr>
<td>Finnish</td>
<td>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz</td>
</tr>
<tr>
<td>Italian</td>
<td>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz</td>
</tr>
<tr>
<td>Norwegian</td>
<td>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz</td>
</tr>
<tr>
<td>Spanish</td>
<td>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz</td>
</tr>
<tr>
<td>Swedish</td>
<td>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz</td>
</tr>
</tbody>
</table>

**Restriction**
You can specify only one collating-sequence-option in a PROC SORT step.

**Tip**
The SORTSEQ= collating sequence options are specified without parenthesis and have no arguments that are associated with them. An example of how to specify a collating sequence follows:

```
proc sort data=mydata SORTSEQ=ASCII;
```

**encoding-value**
specifies an encoding value. The result is the same as a binary collation of the character data represented in the specified encoding. See the supported encoding values in “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 613.

**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 613.

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 559 and “SORT” in SAS Viya Utility Procedures Guide.

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 Utility Procedures Guide

The “Collating Sequence” on page 14 for detailed information about linguistic collation.

“SORTSEQ=sort-table | LINGUISTIC” in SAS Viya SQL Procedure User’s Guide for information about linguistic sorting in PROC SORT.


The following 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>
</tbody>
</table>
STROKE 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.

TRADITIONAL specifies a traditional style for ordering of characters. For example, select TRADITIONAL with the Spanish language.

$LOCALE=locale_name$
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 599 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>
</tbody>
</table>
## Details

The collating sequence option in the PROC SORT statement sorts observations in a SAS data set by one or more characters or numeric variables.

<table>
<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 571</td>
</tr>
</tbody>
</table>
### CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options

Specifies attributes for character variables that are needed in order 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>
```

#### 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 CVPPBYTES= option.

```sas
libname expand 'SAS data-library' cvpbytes=5;
```

Character variable lengths are increased by adding 5 bytes. A character variable with a length of 10 is increased to 15, and a character variable with a length of 100 is increased to 105.

**Default**
If you specify CVPPBYTES=, 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 CVPPBYTES= or CVPMULTIPLIER=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths of the character variables.

**Restrictions**
The CVP engine supports SAS data files, no SAS views, catalogs, item stores, and so on.

The CVP engine is available for input (read) processing only.

For library concatenation with mixed engines that include the CVP engine, only SAS data files are processed. For example, if you execute the COPY procedure, only SAS data files are copied.

**Requirement**
The number of bytes that you specify must be large enough to accommodate any expansion. Otherwise, truncation occurs, which results in an error message in the SAS log.

**Interaction**
You cannot specify both CVPPBYTES= and CVPMULTIPLIER=.
Specify one of these options.

**See**
“Avoiding Character Data Truncation By Using the CVP Engine” on page 23

---

**CVPPENGINE=**

Specifies the engine to use in order to process a SAS data file that requires transcoding. The CVP engine expands the character variable lengths to transcoding so that character data truncation does not occur. Then the specified engine does the actual file processing.

**Alias**
CVPPENG

**Default**
SAS uses the default SAS engine.

**See**
“Avoiding Character Data Truncation By Using the CVP Engine” on page 23

---

**CVPMULTIPLIER=**

Specifies a multiplier value in order to expand character variable lengths when you are processing a SAS data file that requires transcoding. The CVP engine expands the lengths so that character data truncation does not occur. The lengths for character variables are increased by multiplying the current length by the specified value. You can specify a multiplier value from 1 to 5.
For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPMULTIPLIER= option.

```sas
libname expand 'SAS data-library' cvpmultiplier=2.5;
```

Character variable lengths are increased by multiplying the lengths by 2.5. A character variable with a length of 10 is increased to 25, and a character variable with a length of 100 is increased to 250.

### Alias
CVPMULT

### Default
If you specify CVPMULTIPLIER=, SAS automatically uses the CVP engine in order to expand the character variable lengths according to 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

---

### Example: Using the CVP (Character Variable Padding) 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 Wlatin2 encoding.

```sas
libname myfiles 'SAS-Data Library';
data myfiles.wlatin2 (encoding=wlatin2);
  var1='41'x;
  var2='8a'x;
  var3='9c'x;
  var4='b3'x;
;
proc print data=myfiles.wlatin2;
run;
```
Here is PROC CONTENTS output for MYFILES.WLATIN2, which shows that the encoding is Wlatin2 and that the length for each character variable is 1 byte:

Output 18.1 PROC CONTENTS Output for MYFILES.WLATIN2

The following code is executed with the session encoding Wlatin2.

```
options msglevel=i;
libname myfiles 'SAS data-library';
data myfiles.utf8 (encoding="utf-8");
  set myfiles.wlatin2;
run;
```

The DATA step requests a new data set named MYFILES.UTF8, and requests that the data be read into the new data set in UTF-8 encoding, which means that the data must be transcoded from Wlatin2 to UTF-8. The request results in errors due to character data.
truncation that occurs from the transcoding. The new data set MYFILES.UTF8 is created but does not contain any data.

**Log 18.1  SAS Log with Transcoding Error**

```sas
options msglevel=i;
libname myfiles 'C:\Documents and Settings\xxxxxx\My Documents\myfiles';
NOTE: Libref MYFILES was successfully assigned as follows:
   Engine:        V9
   Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
data myfiles.utf8 (encoding="utf-8");
set myfiles.wlatin2;
run;
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 again with the session encoding Wlatin2.

```sas
options msglevel=i;
libname myfiles 'SAS data-library';
libname expand cvp 'SAS data-library' cvpbytes=2;
data myfiles.utf8 (encoding="utf-8");
set expand.wlatin2;
run;
```

In this example, the CVP engine is used to expand character variable lengths by adding two 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 18.2  SAS Log Output for MYFILES.UTF8**

```sas
options msglevel=i;
libname myfiles 'C:\Documents and Settings\xxxxxx\My Documents\myfiles';
NOTE: Libref MYFILES was successfully assigned as follows:
   Engine:        V9
   Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
libname expand cvp 'C:\Documents and Settings\xxxxxx\My Documents\myfiles' cvpbytes=2;
WARNING: Libname EXPAND refers to the same physical library as MYFILES.
NOTE: Libref EXPAND was successfully assigned as follows:
   Engine:        CVP
   Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
data myfiles.utf8 (encoding="utf-8");
set expand.wlatin2;
run;
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.
NOTE: There were 1 observations read from the data set EXPAND.WLATIN2.
NOTE: The data set MYFILES.UTF8 has 1 observations and 4 variables.
```
Finally, here is PROC CONTENTS output for MYFILES.UTF8 showing that it is in UTF-8 encoding and that the length of each character variable is 3:

Output 18.2 PROC CONTENTS Output for MYFILES.UTF8

<table>
<thead>
<tr>
<th>The SAS System</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Name</td>
<td>MYFILES.UTF8</td>
</tr>
<tr>
<td>Observations</td>
<td>1</td>
</tr>
<tr>
<td>Member Type</td>
<td>DATA</td>
</tr>
<tr>
<td>Variables</td>
<td>4</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
</tr>
<tr>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>Thursday, November 07, 2003 02:40:34</td>
</tr>
<tr>
<td>Length</td>
<td>12</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Thursday, November 07, 2003 02:40:34</td>
</tr>
<tr>
<td>Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td>NO</td>
</tr>
<tr>
<td>Compressed</td>
<td>NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td>NO</td>
</tr>
<tr>
<td>Sorted</td>
<td>NO</td>
</tr>
<tr>
<td>Label</td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>WINDOWS_32</td>
</tr>
<tr>
<td>Encoding</td>
<td>utf-8 Unicode (UTF-8)</td>
</tr>
<tr>
<td>Data Set Page Size</td>
<td>4096</td>
</tr>
<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>335</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>File Name</td>
<td>C:\Documents and Settings\xxxxxx\My Documents \myfiles\utf8.sas7bdat</td>
</tr>
<tr>
<td>Release Created</td>
<td>9.0100A0</td>
</tr>
<tr>
<td>Host Created</td>
<td>XP_PRO</td>
</tr>
</tbody>
</table>

Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Var1</td>
<td>Char</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Var2</td>
<td>Char</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Var3</td>
<td>Char</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Var4</td>
<td>Char</td>
<td>3</td>
</tr>
</tbody>
</table>

ENCODING= Option

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 613.

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:

```plaintext
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:

```
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*
Syntax

INENCODING= ANY | ASCIIANY | EBCDICANY | encoding-value
OUTENCODING= ANY | ASCIIANY | EBCDICANY | encoding-value

Syntax Description

ANY
specifies no transcoding between ASCII and EBCDIC encodings.

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

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

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

encoding-value
specifies an encoding value. For a list of encoding values, see Chapter 20, “Encoding Values in SAS Language Elements,” on page 613.

Details

The INENCODING= option is used to read SAS data sets in the SAS library. The OUTENCODING= option is used to write SAS data sets in the SAS library.

The INENCODING= value or the OUTENCODING= value is written to the SAS log when you use the LIST argument.

INENCODING= and OUTENCODING= are most appropriate when using an existing library that contains mixed encodings. To read a library that contains mixed encodings, you can set INENCODING= to ASCIIANY or EBCDICANY. To write a separate data set, you can use OUTENCODING= to specify a specific encoding, which is applied to the data set when it is created.

Comparisons

• Session encoding is specified using the ENCODING= system option.
• You can specify the encoding for reading data sets in a SAS library by using the LIBNAME statement INENCODING= option for input files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.
• You can specify the encoding for writing data sets to a SAS library by using the LIBNAME statement OUTENCODING= option for output files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.
• For the COPY procedure, the default CLONE option uses the encoding attribute of the input data set instead of the encoding value specified on the OUTENCODING= option. For more information about CLONE and NOCLONE, see COPY Statement.

See Also

• “Overview: Encoding for NLS” on page 9
Statements:
• “LIBNAME Statement” in *SAS Viya Statements: Reference*

System Options:
• “ENCODING System Option” on page 549
• “LOCALE System Option” on page 550

Data Set Options:
• “ENCODING= Data Set Option” on page 49

---

**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
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 550

Statements:
• SAS Viya XML LIBNAME Engine: User’s Guide

TRANSCODE= Column Modifier on PROC SQL

Specifies whether values can be transcoded for character columns.

Valid in: Column modifier component in the SQL Procedure
Syntax

TRANSCODE=YES|NO

Required Argument

TRANSCODE=YES|NO

for character columns, specifies whether values can be transcoded. Use TRANSCODE=NO to suppress transcoding. Note that when you create a table using the CREATE TABLE AS statement, the transcoding attribute for a particular character column in the created table is the same as it is in the source table unless you change it with the TRANSCODE= column modifier.

Default YES

Restriction Suppression of transcoding is not supported for the V6TAPE engine.

See Also

• Chapter 4, “Transcoding for NLS,” on page 19
• SAS Viya Utility Procedures Guide

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=NO 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.

You can use the VTRANSCODE and VTRANSCODEX functions to return whether transcoding is on or off for a character variable.

If the TRANSCODE= attribute is set to NO for any character variable in a data set, PROC CONTENTS prints a transcode column that contains the TRANSCODE= value for each variable in the data set. If all variables in the data set are set to the default TRANSCODE= value (YES), no transcode column is printed.

### Examples

**Example 1: Using the TRANSCODE= Option with the SET Statement**

When you use the SET statement to create a data set from several data sets, SAS makes the TRANSCODE= attribute of the variable in the output data set equal to the TRANSCODE= value of the variable in the first data set. In this example, the variable Z’s TRANSCODE= attribute in data set A is NO because B is the first data set and Z’s TRANSCODE= attribute in data set B is NO.

```sas
data b;
  length z $4;
  z = 'ice';
  attrib z transcode = NO;
data c;
  length z $4;
  z = 'snow';
  attrib z transcode = YES;
data a;
  set b;
  set c;
  /* Check transcode setting for variable Z */
  rc1 = vtranscode(z);
  put rc1=;
rung;
```

**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';
```

```sas
data c;
  length z $4;
  z = 'snow';
  attrib z transcode = YES;
data a;
  set b;
  set c;
  /* Check transcode setting for variable Z */
  rc1 = vtranscode(z);
  put rc1=;
rung;
```
attrib z transcode = NO;
data c;
  length z $4;
  z = 'snow';
  attrib z transcode = YES;

data a;
  merge c b;
  /* Check transcode setting for variable Z */
  rc1 = vtranscode(z);
  put rc1=;
run;

Note: The TRANSCODE= attribute is set when the variable is first seen on an input data set or in an ATTRIB TRANSCODE= statement. If a SET or MERGE statement comes before an ATTRIB TRANSCODE= statement and the TRANSCODE= attribute contradicts the SET statement, an error message occurs.

See Also

Functions:
- “VTRANSCODE Function” on page 398
- “VTRANSCODEX Function” on page 399

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 613.

The default for encoding-value is the current session encoding.

See Also

Statements:

• SAS Viya XML LIBNAME Engine: User's Guide

TRANTAB Statement

Specifies the translation table to use when you transcode character data in order to export or transfer a SAS file.

Valid in: CPOR 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 http://support.sas.com/techsup/technote/ts639.pdf

PROC CPORT is used when you transfer a SAS file across a network. PROC UPLOAD and PROC DOWNLOAD are used when you transfer a SAS file across a network.

You must specify the INCAT= and OUTCAT= options in the PROC UPLOAD or PROC DOWNLOAD statement when using the TRANTAB statement.

Examples

Example 1
The information that follows applies to procedure features:

• PROC CPORT statement option: FILE=
• TRANTAB statement option: TYPE=

This example shows how to apply a customized translation table to the transport file before PROC CPORT exports it. For this example, assume that you have already created a customized translation table called TTABLE1.

Example 2: Program
Assign library references. 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 11

Values for Locale and Transcoding

Chapter 19
Values for the LOCALE= System Option .................................. 599

Chapter 20
Encoding Values in SAS Language Elements ................................. 613
Chapter 19
Values for the LOCALE= System Option

The following table lists the valid LOCALE= values, specified by using the SAS name or the POSIX name. The alias name is also listed. Some locales do not have an alias.

**Table 19.1  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_UnitedArab Emirates</td>
<td>ar_AE</td>
<td>Arabic ar</td>
</tr>
<tr>
<td>Arabic_Yemen</td>
<td>ar_YE</td>
<td></td>
</tr>
<tr>
<td>Basque_Spain</td>
<td>eu_ES</td>
<td>Basque Euskara eu</td>
</tr>
<tr>
<td>Bengali_India</td>
<td>bn_IN</td>
<td>Bengali bn</td>
</tr>
<tr>
<td>Bosnian_BosniaHerzegovina</td>
<td>bs_BA</td>
<td>Bosnian bs</td>
</tr>
<tr>
<td>Bulgarian_Bulgaria</td>
<td>bg_BG</td>
<td>Bulgarian bg</td>
</tr>
<tr>
<td>Byelorussian_Belarus</td>
<td>be_BY</td>
<td>Byelorussian Belarusian Byelorussian_Belarus be</td>
</tr>
<tr>
<td>Catalan_Spain</td>
<td>ca_ES</td>
<td>Catalan ca</td>
</tr>
<tr>
<td>Chinese_China</td>
<td>zh_CN</td>
<td>Chinese 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 kl</td>
</tr>
<tr>
<td>Hebrew_Israel</td>
<td>he_IL</td>
<td>Hebrew he</td>
</tr>
<tr>
<td>Hindi_India</td>
<td>hi_IN</td>
<td>Hindi hi</td>
</tr>
<tr>
<td>Hungarian_Hungary</td>
<td>hu_HU</td>
<td>Hungarian hu</td>
</tr>
<tr>
<td>Icelandic_Iceland</td>
<td>is_IS</td>
<td>Icelandic is</td>
</tr>
<tr>
<td>Indonesian_Indonesia</td>
<td>id_ID</td>
<td>Indonesian id</td>
</tr>
<tr>
<td>Irish_Ireland</td>
<td>ga_IE</td>
<td>Irish ga</td>
</tr>
<tr>
<td>Italian_Italy</td>
<td>it_IT</td>
<td>Italian 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 ja</td>
</tr>
<tr>
<td>Kazakh_Kazakhstan</td>
<td>kk_KZ</td>
<td>Kazakh kk</td>
</tr>
<tr>
<td>Korean_Korea</td>
<td>ko_KR</td>
<td>Korean ko</td>
</tr>
<tr>
<td>Korean_SouthKorea</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latvian_Latvia</td>
<td>lv_LV</td>
<td>Latvian lv</td>
</tr>
<tr>
<td>Lithuanian_Lithuania</td>
<td>lt_LT</td>
<td>Lithuanian lt</td>
</tr>
<tr>
<td>Macedonian_Macedonia</td>
<td>mk_MK</td>
<td>Macedonian mk</td>
</tr>
<tr>
<td>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>Russian ru</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>Serbian sr</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></td>
</tr>
<tr>
<td>SerbianLatin_Serbia</td>
<td>sh_RS</td>
<td>SerbianLatin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sh</td>
</tr>
<tr>
<td>Slovak_Slovakia</td>
<td>sk_SK</td>
<td>Slovak</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slovakian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slovakian_Slovakia</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sk</td>
</tr>
<tr>
<td>Slovenian_Slovenia</td>
<td>sl_SI</td>
<td>Slovenian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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:

<table>
<thead>
<tr>
<th>DFLANG=</th>
<th>DATESTYLE=</th>
<th>PAPERSIZE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
</tbody>
</table>

### Table 19.2
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>letter</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>acyrillic</td>
<td>acyr</td>
<td>MacOS 7 Cyrillic</td>
<td>1</td>
</tr>
<tr>
<td>agreeek</td>
<td>agrk</td>
<td>Greek Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>ahebrew</td>
<td>aheb</td>
<td>Hebrew Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aiceland</td>
<td>aice</td>
<td>Icelandic Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>any</td>
<td>anye</td>
<td>no transcoding is specified</td>
<td>1</td>
</tr>
<tr>
<td>arabic</td>
<td>arab</td>
<td>Arabic ISO</td>
<td>1</td>
</tr>
<tr>
<td>aroman</td>
<td>arom</td>
<td>Roman Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aromania</td>
<td>arma</td>
<td>MacOS 38 Romania</td>
<td>1</td>
</tr>
<tr>
<td>athai</td>
<td>atha</td>
<td>MacOS 21-Thai</td>
<td>1</td>
</tr>
<tr>
<td>aturkish</td>
<td>atur</td>
<td>Turkish Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aukrainian</td>
<td>aukr</td>
<td>Ukrainian Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>big5</td>
<td>big5</td>
<td>Traditional Chinese Big5</td>
<td>2</td>
</tr>
<tr>
<td>cyrillic</td>
<td>cyrl</td>
<td>Cyrillic ISO</td>
<td>1</td>
</tr>
<tr>
<td>dec-cn</td>
<td>zvms</td>
<td>Simplified Chinese DEC</td>
<td>4</td>
</tr>
<tr>
<td>dec-jp</td>
<td>jvms</td>
<td>Japanese DEC</td>
<td>2</td>
</tr>
<tr>
<td>dec-tw</td>
<td>yvms</td>
<td>Traditional Chinese DEC</td>
<td>4</td>
</tr>
<tr>
<td>ebcdic037</td>
<td>e037</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic275</td>
<td>e275</td>
<td>Brazil EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic424</td>
<td>e424</td>
<td>Hebrew EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic425</td>
<td>e425</td>
<td>Arabic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic500</td>
<td>e500</td>
<td>International EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic838</td>
<td>e838</td>
<td>Thai EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>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>ebcdic870</td>
<td>e870</td>
<td>Central European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic875</td>
<td>e875</td>
<td>Greek EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic905</td>
<td>e905</td>
<td>Latin 3 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic924</td>
<td>e924</td>
<td>European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1025</td>
<td>ecyr</td>
<td>Cyrillic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1026</td>
<td>etur</td>
<td>Turkish EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1047</td>
<td>elat</td>
<td>Western EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1097</td>
<td>e097</td>
<td>Farsi Bilingual EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1112</td>
<td>ebal</td>
<td>Baltic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1122</td>
<td>eest</td>
<td>Estonian EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1130</td>
<td>evie</td>
<td>Vietnamese EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1137</td>
<td>e137</td>
<td>Devanagari EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1140</td>
<td>e140</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1141</td>
<td>e141</td>
<td>Austria/Germany EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1142</td>
<td>e142</td>
<td>Denmark/Norway EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1143</td>
<td>e143</td>
<td>Finland/Sweden EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1144</td>
<td>e144</td>
<td>Italy EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1145</td>
<td>e145</td>
<td>Spain EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1146</td>
<td>e146</td>
<td>United Kingdom EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1147</td>
<td>e147</td>
<td>France EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1148</td>
<td>e148</td>
<td>International EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>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 dicany</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>jhxs</td>
<td>Japanese XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-ko</td>
<td>khxs</td>
<td>Korean XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-tw</td>
<td>yhxs</td>
<td>Traditional Chinese XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hp15-tw</td>
<td>yhpx</td>
<td>Traditional Chinese HP15</td>
<td>2</td>
</tr>
<tr>
<td>ibm-1381</td>
<td>zpce</td>
<td>Simplified Chinese PCIBM</td>
<td>2</td>
</tr>
<tr>
<td>ibm-930</td>
<td>j930</td>
<td>Japanese katakana</td>
<td>4</td>
</tr>
<tr>
<td>ibm-933</td>
<td>kibm</td>
<td>Korean IBM</td>
<td>4</td>
</tr>
<tr>
<td>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 OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-424</td>
<td>eoiw</td>
<td>Hebrew OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-425</td>
<td>eoa2</td>
<td>Arabic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-838</td>
<td>eoth</td>
<td>Thai OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-870</td>
<td>eol2</td>
<td>Central European OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-875</td>
<td>eoel</td>
<td>Greek OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-905</td>
<td>eol3</td>
<td>Latin 3 Open Edition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-924</td>
<td>eolt</td>
<td>European OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-930</td>
<td>oe30</td>
<td>Katakana OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-933</td>
<td>oe33</td>
<td>Korean OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-935</td>
<td>oe35</td>
<td>Simplified Chinese OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-937</td>
<td>oe37</td>
<td>Traditional Chinese OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-939</td>
<td>oe39</td>
<td>Japanese IBM</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-939E</td>
<td>oe9e</td>
<td>Japanese IBM Extended’</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-1025</td>
<td>eocy</td>
<td>Cyrillic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1026</td>
<td>eotr</td>
<td>Turkish OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1047</td>
<td>eol1</td>
<td>Western OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1097</td>
<td>eofa</td>
<td>Farsi Bilingual OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1112</td>
<td>eobl</td>
<td>Baltic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>open_ed-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>wrkr</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.
Part 12

Appendixes

Appendix 1

*Time Zone IDs and Time Zone Names* .......................... 625
Appendix 1
Time Zone IDs and Time Zone Names

<table>
<thead>
<tr>
<th>Area: Africa</th>
<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></td>
<td>Africa/Abidjan</td>
<td>GMT</td>
<td>CI</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>Africa/Accra</td>
<td>GMT</td>
<td>GH</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>Africa/Bamako</td>
<td>GMT</td>
<td>ML</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>Africa/Banjul</td>
<td>GMT</td>
<td>GM</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>Africa/Bissau</td>
<td>GMT</td>
<td>GW</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>Africa/Conakry</td>
<td>GMT</td>
<td>GN</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>Africa/Dakar</td>
<td>GMT</td>
<td>SN</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>Africa/Freetown</td>
<td>GMT</td>
<td>SL</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<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>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<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>
<td></td>
</tr>
<tr>
<td>Africa/Nouakchott</td>
<td>GMT</td>
<td>MR</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Ouagadougou</td>
<td>GMT</td>
<td>BF</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Sao_Tome</td>
<td>GMT</td>
<td>ST</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Timbuktu</td>
<td>GMT</td>
<td>ML</td>
<td>Greenwich Mean Time</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Africa/Casablanca</td>
<td>WET</td>
<td>MA</td>
<td>Western European Time</td>
<td>00:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEST</td>
<td>MA</td>
<td>Western European Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Els_Aaiun</td>
<td>WET</td>
<td>EH</td>
<td>Western European Time</td>
<td>00:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEST</td>
<td>EH</td>
<td>Western European Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Algiers</td>
<td>CET</td>
<td>DZ</td>
<td>Central European Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Ceuta</td>
<td>CET</td>
<td>ES</td>
<td>Central European Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CEST</td>
<td></td>
<td>Central European Summer Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Tunis</td>
<td>CET</td>
<td>TN</td>
<td>Central European Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Bangui</td>
<td>WAT</td>
<td>CF</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Brazzaville</td>
<td>WAT</td>
<td>CG</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Douala</td>
<td>WAT</td>
<td>CM</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Kinshasa</td>
<td>WAT</td>
<td>CD</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Lagos</td>
<td>WAT</td>
<td>NG</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Libreville</td>
<td>WAT</td>
<td>GA</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Luanda</td>
<td>WAT</td>
<td>AO</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Malabo</td>
<td>WAT</td>
<td>GQ</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Ndjamea</td>
<td>WAT</td>
<td>TD</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Niamey</td>
<td>WAT</td>
<td>NE</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Porto-Nov</td>
<td>WAT</td>
<td>BJ</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Windhoek</td>
<td>WAT</td>
<td>NA</td>
<td>West Africa Time</td>
<td>01:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WAST</td>
<td></td>
<td>West Africa Summer Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Blantyre</td>
<td>CAT</td>
<td>MW</td>
<td>Central Africa Time</td>
<td>02:00</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>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<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>
<td></td>
</tr>
<tr>
<td>Africa/Gaborone</td>
<td>CAT</td>
<td>BW</td>
<td>Central Africa Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Harare</td>
<td>CAT</td>
<td>ZW</td>
<td>Central Africa Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Kigali</td>
<td>CAT</td>
<td>RW</td>
<td>Central Africa Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Lubumbashi</td>
<td>CAT</td>
<td>CD</td>
<td>Central Africa Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Lusaka</td>
<td>CAT</td>
<td>ZM</td>
<td>Central Africa Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Maputo</td>
<td>CAT</td>
<td>MZ</td>
<td>Central Africa Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Cairo</td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EET</td>
<td>EG</td>
<td>Eastern European</td>
<td>2:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Tripoli</td>
<td>EET</td>
<td>LY</td>
<td>Eastern European Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Johannesburg</td>
<td>SAST</td>
<td>ZA</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Maseru</td>
<td>SAST</td>
<td>LS</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Mbabane</td>
<td>SAST</td>
<td>SZ</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Addis_Ababa</td>
<td>EAT</td>
<td>ET</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Asmara</td>
<td>EAT</td>
<td>ER</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Asmera</td>
<td>EAT</td>
<td>ER</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Dar_es_Salaam</td>
<td>EAT</td>
<td>TZ</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Djibouti</td>
<td>EAT</td>
<td>DJ</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Juba</td>
<td>EAT</td>
<td>SS</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Kampala</td>
<td>EAT</td>
<td>UG</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Khartoum</td>
<td>EAT</td>
<td>SD</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Mogadishu</td>
<td>EAT</td>
<td>SO</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td>Africa/Nairobi</td>
<td>EAT</td>
<td>KE</td>
<td>East Africa Time</td>
<td>03:00</td>
<td></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</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Goose_Bay</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EST</td>
<td>TC</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EDT</td>
<td>TC</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Grenada</td>
<td>AST</td>
<td>GD</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Guadeloupe</td>
<td>AST</td>
<td>GP</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Guadeloupe</td>
<td>EST</td>
<td>GP</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Guatemala</td>
<td>CST</td>
<td>GT</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Guayaquil</td>
<td>ECT</td>
<td>EC</td>
<td>Ecuador Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Guyana</td>
<td>GYT</td>
<td>GY</td>
<td>Guyana Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Halifax</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Halifax</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Havana</td>
<td>CST</td>
<td>CU</td>
<td>Cuba Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Havana</td>
<td>CDT</td>
<td>CU</td>
<td>Cuba Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Hermosillo</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indiana/Vincennes</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Vincennes</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>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/Santararem</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 Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Winnipeg</td>
<td>CDT</td>
<td>CA</td>
<td>Central Standard 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/Novokuzeznetsk</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>
</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>
</tbody>
</table>
## Time Zone Information

<table>
<thead>
<tr>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlantic/Faroe</td>
<td>WET</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Faroe</td>
<td>WEST</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CET</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CEST</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WET</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WEST</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Reykjavik</td>
<td>GMT</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/South_Georgia</td>
<td>GST</td>
<td>South Georgia Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Atlantic/St_Helena</td>
<td>GMT</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Stanley</td>
<td>FKST</td>
<td>Falkland Islands Standard Time</td>
<td>-3:00</td>
</tr>
</tbody>
</table>

## Area: Australia

<table>
<thead>
<tr>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia/ACT</td>
<td>EST</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/ACT</td>
<td>EDT</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>CST</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>CDT</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Brisbane</td>
<td>EST</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>CST</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>CDT</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>EST</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>EDT</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>EST</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>EDT</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Darwin</td>
<td>CST</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>
</tr>
<tr>
<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>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>LHST</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>LHDT</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
</tr>
<tr>
<td>Australia/Lindeman</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>LHST</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>LHDT</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
</tr>
<tr>
<td>Australia/North</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
</tr>
<tr>
<td>Australia/Perth</td>
<td>WST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
</tr>
<tr>
<td>Australia/Queensland</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/South</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
</tr>
<tr>
<td>Australia/South</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
</tr>
<tr>
<td>Australia/West</td>
<td>WST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</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></td>
<td>1:00</td>
</tr>
<tr>
<td>Etc/GMT</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/GMT+0</td>
<td>GMT</td>
<td>IE</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>IE</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 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>Hongkong</td>
<td>HKT</td>
<td>HK</td>
<td>Hong Kong Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Iceland</td>
<td>GMT</td>
<td>IS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Indian/Antananarivo</td>
<td>EAT</td>
<td>MG</td>
<td>East Africa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Indian/Chagos</td>
<td>IOT</td>
<td>IO</td>
<td>Indian Ocean Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Indian/Christmas</td>
<td>CXT</td>
<td>CX</td>
<td>Christmas Island Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Indian/Cocos</td>
<td>CCT</td>
<td>CC</td>
<td>Cocos Islands Time</td>
<td>6:30</td>
</tr>
<tr>
<td>Indian/Comoro</td>
<td>EAT</td>
<td>KM</td>
<td>East Africa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Indian/Kerguelen</td>
<td>TFT</td>
<td>TF</td>
<td>French Southern and Antarctic Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Indian/Mahe</td>
<td>SCT</td>
<td>SC</td>
<td>Seychelles Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Indian/Maldives</td>
<td>MVT</td>
<td>MV</td>
<td>Maldives Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Indian/Mauritius</td>
<td>MUT</td>
<td>MU</td>
<td>Mauritius Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Indian/Mayotte</td>
<td>EAT</td>
<td>YT</td>
<td>East Africa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Indian/Reunion</td>
<td>RET</td>
<td>RE</td>
<td>Reunion Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Iran</td>
<td>IRST</td>
<td>IR</td>
<td>Iran Standard Time</td>
<td>3:30</td>
</tr>
<tr>
<td>Israel</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Israel</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Jamaica</td>
<td>EST</td>
<td>JM</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Japan</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Kwajalein</td>
<td>MHT</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Libya</td>
<td>EET</td>
<td>LY</td>
<td>Central European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MST</td>
<td></td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MDT</td>
<td></td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Mexico/General</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Mexico/General</td>
<td>CDT</td>
<td>MX</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>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>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>PST8PDT</td>
<td>PDT</td>
<td>US</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></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>
</tbody>
</table>
### Time Zone Information

<table>
<thead>
<tr>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>US/Eastern</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
</tr>
<tr>
<td>US/Eastern</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
</tr>
<tr>
<td>US/Hawaii</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
</tr>
<tr>
<td>US/Samoa</td>
<td>SST</td>
<td>AS</td>
<td>Samoa Standard Time</td>
</tr>
<tr>
<td>UTC</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Universal</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>W-SU</td>
<td>MSK</td>
<td>RU</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Zulu</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
</tr>
</tbody>
</table>

### Area: Europe

<table>
<thead>
<tr>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe/Amsterdam</td>
<td>CET</td>
<td>NL</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Amsterdam</td>
<td>CEST</td>
<td>NL</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CET</td>
<td>AD</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CEST</td>
<td>AD</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Athens</td>
<td>EET</td>
<td>GR</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Europe/Athens</td>
<td>EEST</td>
<td>GR</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>BST</td>
<td>GB</td>
<td></td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CET</td>
<td>RS</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CEST</td>
<td>RS</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CET</td>
<td>SK</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CEST</td>
<td>SK</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CET</td>
<td>BE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CEST</td>
<td>BE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EET</td>
<td>RO</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EEST</td>
<td>RO</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CET</td>
<td>HU</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CEST</td>
<td>HU</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CET</td>
<td>DK</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CEST</td>
<td>DK</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>IST</td>
<td>IE</td>
<td></td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CET</td>
<td>GI</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CEST</td>
<td>GI</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>GMT</td>
<td>GG</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>BST</td>
<td>GG</td>
<td></td>
</tr>
<tr>
<td>Europe/Helsinki</td>
<td>EET</td>
<td>FI</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Europe/Helsinki</td>
<td>EEST</td>
<td>FI</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>GMT</td>
<td>IM</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>BST</td>
<td>IM</td>
<td></td>
</tr>
<tr>
<td>Europe/Istanbul</td>
<td>EET</td>
<td>TR</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Istanbul</td>
<td>EEST</td>
<td>TR</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>GMT</td>
<td>JE</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>BST</td>
<td>JE</td>
<td></td>
</tr>
<tr>
<td>Europe/Kaliningrad</td>
<td>FET</td>
<td>RU</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CET</td>
<td>SI</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CEST</td>
<td>SI</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/London</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>Europe/London</td>
<td>BST</td>
<td>GB</td>
<td></td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CET</td>
<td>LU</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CEST</td>
<td>LU</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CET</td>
<td>ES</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CEST</td>
<td>ES</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CET</td>
<td>MT</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CEST</td>
<td>MT</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EET</td>
<td>AX</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EEST</td>
<td>AX</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Minsk</td>
<td>FET</td>
<td>BY</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CET</td>
<td>MC</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CEST</td>
<td>MC</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Moscow</td>
<td>MSK</td>
<td>RU</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Europe/Nicosia</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Nicosia</td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Oslo</td>
<td>CET</td>
<td>NO</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Oslo</td>
<td>CEST</td>
<td>NO</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Paris</td>
<td>CET</td>
<td>FR</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Paris</td>
<td>CEST</td>
<td>FR</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Podgorica</td>
<td>CET</td>
<td>ME</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Podgorica</td>
<td>CEST</td>
<td>ME</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Prague</td>
<td>CET</td>
<td>CZ</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Prague</td>
<td>CEST</td>
<td>CZ</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Riga</td>
<td>EET</td>
<td>LV</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Riga</td>
<td>EEST</td>
<td>LV</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Rome</td>
<td>CET</td>
<td>IT</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Rome</td>
<td>CEST</td>
<td>IT</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Samara</td>
<td>SAMT</td>
<td>RU</td>
<td>Moscow Standard Time</td>
</tr>
<tr>
<td>Europe/San_Marino</td>
<td>CET</td>
<td>SM</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/San_Marino</td>
<td>CEST</td>
<td>SM</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CET</td>
<td>BA</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CEST</td>
<td>BA</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Simferopol</td>
<td>MSK</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CET</td>
<td>MK</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CEST</td>
<td>MK</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EET</td>
<td>BG</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EEST</td>
<td>BG</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CET</td>
<td>SE</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CEST</td>
<td>SE</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EET</td>
<td>EE</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EEST</td>
<td>EE</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Europe/Tirane</td>
<td>CET</td>
<td>AL</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Tirane</td>
<td>CEST</td>
<td>AL</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CET</td>
<td>LI</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CEST</td>
<td>LI</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CET</td>
<td>VA</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CEST</td>
<td>VA</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CET</td>
<td>AT</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CEST</td>
<td>AT</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EET</td>
<td>LT</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EEST</td>
<td>LT</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Volgograd</td>
<td>VOLT</td>
<td>RU</td>
<td>Volgograd Standard Time</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CET</td>
<td>PL</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CEST</td>
<td>PL</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CET</td>
<td>HR</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CEST</td>
<td>HR</td>
<td>Central European Summer Time</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CET</td>
<td>CH</td>
<td>Central European Standard Time</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CEST</td>
<td>CH</td>
<td>Central European Summer Time</td>
</tr>
</tbody>
</table>
## Area: Pacific

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pacific/Apia</td>
<td>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/Efate</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/Kiritimati</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 Utility Procedures Guide*
- *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)
### Index

#### Special Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Format/Informat</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CPTDW</td>
<td>format</td>
<td>90</td>
</tr>
<tr>
<td>SCPTWD</td>
<td>format</td>
<td>91</td>
</tr>
<tr>
<td>SEBCDICw.</td>
<td>format</td>
<td>92</td>
</tr>
<tr>
<td>SUCS2B</td>
<td>format</td>
<td>243</td>
</tr>
<tr>
<td>SUCS2B</td>
<td>informat</td>
<td>511</td>
</tr>
<tr>
<td>SUCS2BE</td>
<td>format</td>
<td>245</td>
</tr>
<tr>
<td>SUCS2BE</td>
<td>informat</td>
<td>512</td>
</tr>
<tr>
<td>SUCS2L</td>
<td>format</td>
<td>246</td>
</tr>
<tr>
<td>SUCS2L</td>
<td>informat</td>
<td>513</td>
</tr>
<tr>
<td>SUCS2LE</td>
<td>format</td>
<td>247</td>
</tr>
<tr>
<td>SUCS2LE</td>
<td>informat</td>
<td>514</td>
</tr>
<tr>
<td>SUCS2X</td>
<td>format</td>
<td>248</td>
</tr>
<tr>
<td>SUCS2X</td>
<td>informat</td>
<td>515</td>
</tr>
<tr>
<td>SUCS2XE</td>
<td>format</td>
<td>250</td>
</tr>
<tr>
<td>SUCS2XE</td>
<td>informat</td>
<td>516</td>
</tr>
<tr>
<td>SUCS4B</td>
<td>format</td>
<td>251</td>
</tr>
<tr>
<td>SUCS4B</td>
<td>informat</td>
<td>517</td>
</tr>
<tr>
<td>SUCS4BE</td>
<td>format</td>
<td>252</td>
</tr>
<tr>
<td>SUCS4L</td>
<td>format</td>
<td>253</td>
</tr>
<tr>
<td>SUCS4L</td>
<td>informat</td>
<td>518</td>
</tr>
<tr>
<td>SUCS4LE</td>
<td>format</td>
<td>255</td>
</tr>
<tr>
<td>SUCS4X</td>
<td>format</td>
<td>256</td>
</tr>
<tr>
<td>SUCS4X</td>
<td>informat</td>
<td>519</td>
</tr>
<tr>
<td>SUCS4XE</td>
<td>format</td>
<td>257</td>
</tr>
<tr>
<td>SUCS4XE</td>
<td>informat</td>
<td>521</td>
</tr>
<tr>
<td>SUESC</td>
<td>format</td>
<td>258</td>
</tr>
<tr>
<td>SUESC</td>
<td>informat</td>
<td>522</td>
</tr>
<tr>
<td>SUESC</td>
<td>format</td>
<td>259</td>
</tr>
<tr>
<td>SUESC</td>
<td>informat</td>
<td>523</td>
</tr>
<tr>
<td>SUNC</td>
<td>format</td>
<td>260</td>
</tr>
<tr>
<td>SUNC</td>
<td>informat</td>
<td>524</td>
</tr>
<tr>
<td>SUNCRE</td>
<td>format</td>
<td>261</td>
</tr>
<tr>
<td>SUNCRE</td>
<td>informat</td>
<td>525</td>
</tr>
<tr>
<td>SUPAREN</td>
<td>format</td>
<td>262</td>
</tr>
<tr>
<td>SUPAREN</td>
<td>informat</td>
<td>526</td>
</tr>
<tr>
<td>SUPARENE</td>
<td>format</td>
<td>264</td>
</tr>
<tr>
<td>SUPARENE</td>
<td>informat</td>
<td>527</td>
</tr>
<tr>
<td>SUPARENP</td>
<td>format</td>
<td>265</td>
</tr>
<tr>
<td>SUPARENP</td>
<td>informat</td>
<td>528</td>
</tr>
<tr>
<td>SUTF8X</td>
<td>format</td>
<td>265</td>
</tr>
<tr>
<td>SUTF8X</td>
<td>informat</td>
<td>530</td>
</tr>
<tr>
<td>SUTF8XE</td>
<td>format</td>
<td>266</td>
</tr>
<tr>
<td>SUTF8XE</td>
<td>informat</td>
<td>534</td>
</tr>
<tr>
<td>%KLEFT</td>
<td>macro function</td>
<td>535</td>
</tr>
<tr>
<td>%KLENGTH</td>
<td>macro function</td>
<td>535</td>
</tr>
<tr>
<td>%KLOWCASE</td>
<td>autocall macro</td>
<td>43</td>
</tr>
<tr>
<td>%KSCAN</td>
<td>macro function</td>
<td>536</td>
</tr>
<tr>
<td>%KSUBSTR</td>
<td>macro function</td>
<td>538</td>
</tr>
<tr>
<td>%KTRIM</td>
<td>autocall macro</td>
<td>44</td>
</tr>
<tr>
<td>%KVERIFY</td>
<td>autocall macro</td>
<td>44</td>
</tr>
<tr>
<td>%QCMPRES</td>
<td>534</td>
<td></td>
</tr>
<tr>
<td>%QKTRIM</td>
<td>autocall macro</td>
<td>44</td>
</tr>
</tbody>
</table>

#### Numbers

- 8859 ISO family | 11

#### A

- alignment
  - character expressions | 382, 386
  - ANORM420 Function | 308
- ANSI (American National Standards Institute) | 13
- arguments
  - converting to lowercase | 384
  - converting to uppercase | 393
  - extracting a substring from | 390
  - extracting a substring from, based on byte position | 330
- length of | 383
- transcoding for specified argument | 399
- ASCII | 11
  - transferring data between EBCDIC and | 22
- ATTRIB statement | 589
  - TRANSCODE= option | 590
- Australia
  - monetary format | 154, 191
- Austria
  - monetary format | 201

#### B

- B8601DXw. format | 84
- B8601LXw. format | 86
- B8601TXw. format | 87
BASECHAR function 376
Belgium
   monetary format 164, 201
BESTDOTX format 88
binary collation 14
blanks
   removing leading DBCS blanks 382
   trimming trailing 44
   trimming trailing DBCS blanks 386
BOMFILE system option 546
Brazil
   monetary format 156, 193
Bulgaria
   monetary format 155, 192
Byte Order Mark (BOM) prefix
   on Unicode external files 546

C
Canada
   monetary format 157, 194
case
   changing uppercase characters to lowercase 43
character data
   converting to EBCDIC 92
character expressions
   comparing 319
   compressing 378
   concatenating 388
deleting character value contents 394
deleting character value contents, based on byte unit 332
inserting character value contents 394
inserting character value contents, based on byte unit 332
left-aligning 382
number of double-byte characters in position of first unique character 397
removing trailing blanks and SO/SI 392
replacing character value contents 394
replacing character value contents, based on byte unit 332
reversing 385
right-aligning 386
searching for specific characters 381
searching for string of characters 380
selecting a specified word from trimming 387
   updating 394
   updating, based on byte unit 332
   verifying 397
character sets
   definition 9
   specifying, for META declaration in output 570
character strings 538
character variables
   transcoding enabled for specified variable 398
character-set encoding
   for SAS session 549
characters
   locating 534
CHARSET= option 570
China
   monetary format 159, 196
code page 9
collating sequence 14
   binary collation 14
   language-specific 559
   linguistic collation 15
   overview 14
   results of different sequences 14
compatibility
   SAS string functions 281
   compatible encodings 21
concatenation
   character expressions 388
   CPTDW informat 414
   CPTWD informat. 415
Croatia
   monetary format 167, 204
currency 6
   yen 271
CVPBYTES option 577
Czech Republic
   monetary format 160, 197

D
data set options
   for transcoding 20
data sets
   encoding 16
   encoding support, by release 17
   mixed encodings 51
   suppressing transcoding 51
   transcoding enabled for specified variable 398
   with a particular encoding 51
date format descriptors 335
date values
   as a date 107
   converting to specified locale 335
   date and day of week 115
day of week 116
Japanese 104, 421
   name and day of month 110
   name of month 113
Taiwanese 102, 419
Index  661

year  126
year and name of month  118
year and quarter  122
year and week  127
dates  5
date values as  107
DATESTYLE= system option
default values  606
datetime values
as a datetime  128
B8601DT format, ISO 8601 basic
notation, no time zone  84, 86
B8601TX format, ISO 8601 basic
notation, time zone offset  87
converting to specified locale  338, 340
day of week  141
day of week and datetime  140
E8601DT format, ISO 8601 basic
notation, no time zone  93, 94
E8601TX format, ISO 8601 basic
notation, time zone offset  96
name and day of month  132
name of month  136
name of month, day of month and year  130
time of day  138
with a.m. or p.m.  129
year  150
year and name of month  143
year and name of week  151
year and quarter  146
datetime-format descriptors  338
DBCS encoding
character data truncation  23
encoding values for transcoding data  613
removing leading blanks  382
trimming blanks and SO/SI  392
trimming trailing blanks  386
Denmark
monetary format  161, 198
DFLANG= system option  547
default values  606
double-byte characters
number in a character expression  379
converting character data to  92
transferring data between ASCII and  22
Egypt
monetary format  163, 200
ENCODCOMPAT function  313
encoding  9
behavior in SAS sessions  17
compatibility for transcoding  21
converting one type of data to another  320
data set support by release  17
data sets  16
definition  10
for East Asian languages  11
input processing  17
mixed  51
output processing  17
overriding  49
reading and writing external files  17
standards organizations for  13
versus transcoding  10
encoding methods  9, 10
ENCODING option  582
ENCODING system option  549
encoding values  613
DBCS  613
for transcoding data  613
SBCS  613
Unicode  613
ENCODING= data set option  49
ENCODING= system option
default settings  606
Posix values  606
ENCODISVALID Function  314
Estonia
monetary format  162, 199
EURO format  97
EURO informat  416
euros
formats for  97, 100
EUROX format  100
EUROX informat  418
external files
BOM prefix on Unicode files  546
encoding and  17

E
E8601DXw. format  93
E8601LXw. format  94
E8601TXw. format  96
East Asian languages
encodings for  11
EBCDIC  11
code point discrepancies among
encodings  13

F
Faroe Island
monetary format  161, 198
filerefs
limiting to eight characters  539
Finland
monetary format  164, 201
formats
associating with variables  589
international date and datetime formats 55
language for international dates 547
NLS 55
France
monetary format 164, 201
functions
by category 306
K functions 281
SAS string functions 281

G
Germany
monetary format 164, 201
GETLOCENV function 315
GETPXLANGUAGE function 316
GETPXLOCAL function 317
GETPXREGION function 318
Greece
monetary format 164, 201
Greenland
monetary format 161, 198

H
Hebrew characters 90, 91
Hong Kong
monetary format 166, 203
Hungary
monetary format 168, 205

I
IBw.d informat 501, 502, 504, 505
illegal data 558
incompatible encodings 21
India
monetary format 171, 208
Indonesia
monetary format 169, 206
INENCODING= and OUTENCODING= option 585
informat
associating with variables 589
language for international dates 547
input processing 17
integer binary values, reading 501, 502, 504, 505
international date and datetime formats 55
international date formats and informats specifying language for 547
International Organization for Standardization (ISO) 13
internationalization 4

Ireland
monetary format 164, 201
ISO (International Organization for Standardization) 13
ISO 8601 date and time formats
B8601DX format, basic datetime notation, time zone offset 84
B8601LX format, basic datetime notation, time zone offset 86
B8601TX format, basic time notation, time zone offset 87
E8601DX format, basic datetime notation, time zone offset 93
E8601LX format, basic datetime notation, time zone offset 94
E8601TX format, basic time notation, time zone offset 96
ISO encodings 11
8859 family 11
Windows family 11
Israel
monetary format 170, 207
Italy
monetary format 164, 201
item stores
naming 561

J
Japan
monetary format 172, 209
Japanese dates 104, 421

K
K functions 281
KCOMPARE function 319
KCOMPRESS function 378
KCOUNT function 379
KCVT function 320
KINDEX function 380
KINDEXB function 322
KINDEXC function 381
KINDEXCB function 323
KLEFT function 382
KLENGTH function 383
KLOWCASE function 384
KPROPCASE function
KPROPCHAR function 327
KPROPDATA function 328
KREVERSE function 385
KRIGHT function 386
KSCAN function 387
KSTRCAT function 388
KSTRIP function 389
KSUBSTR function 390
KSUBSTRB function 330
KTRIM function 392
KTRUNCATE function 331
KUCASE function 393
KUCASE macro function 540
KUPDATE function 394
KUPDATEB function 332
KUPDATES function 395
KVERIFY function 397
KVERIFYB function 334
LOGLANGENG system option 553
LOGLANGGCHG system option 552
LOGLANGCHG system option 552
LOGLANGGCHG system option 553
long macro variables
  storing values in segments 539
lowercase characters
  changing uppercase characters to 43
  converting arguments to 384
LSWL.lang option 554
Luxembourg
  monetary format 164, 201

M
Macau
  monetary format 176, 213
macro variables
  storing long values in segments 539
Malaysia
  monetary format 178, 215
Malta
  monetary format 164, 201
MAPEBCDIC2ASCII system option 555
META declaration 570
Mexico
  monetary format 177, 214
MINGUO format 102
MINGUO informat 419
monetary formats
  Australia 154, 191
  Austria 201
  Belgium 164, 201
  Brazil 156, 193
  Bulgaria 155, 192
  Canada 157, 194
  China 159, 196
  Croatia 167, 204
  Czech Republic 160, 197
  Denmark 161, 198
  Egypt 163, 200
  Estonia 162, 199
  Faroe Island 161, 198
  Finland 164, 201
  France 164, 201
  Germany 164, 201
  Greece 164, 201
  Greenland 161, 198
  Hong Kong 166, 203
  Hungary 168, 205
  India 171, 208
  Indonesia 169, 206
  Ireland 164, 201
  Israel 170, 207
  Italy 164, 201
  Japan 172, 209
  Latvia 175, 212
  Liechtenstein 158, 195
  Lithuania 174, 211

L
labels, associating with variables 589
language 5
language codes
  current two-letter code 316
language switching 7
  changing text language of ODS output 557
languages
  for international date informats and formats 547
Latvia
  monetary format 175, 212
left-aligning character expressions 382
length
  associating with variables 589
  of arguments 383
Liechtenstein
  monetary format 158, 195
line-feed characters 22
linguistic collation 15
linguistic sort keys 347
Lithuania
  monetary format 174, 211
locale 5
  best numerical notation based on 105
  converting date values to specified locale 335
  converting datetime values to specified locale 338
  converting time or datetime values to specified locale 340
language switching 7
  of SAS sessions 550
POSIX value for 317
specifying 6
  specifying at SAS invocation 6
  specifying during SAS session 7
LOCALE system option 550
LOCALE= system option
  values for 599
LOCALEDATA system option 551
localization 4
LOGLANGCHG system option 552
Luxembourg 164, 201
Macau 176, 213
Malaysia 178, 215
Malta 164, 201
Mexico 177, 214
Netherlands 164, 201
New Zealand 180, 217
Norway 179, 216
Poland 181, 201
Portugal 164, 201
Puerto Rico 188, 225
Russia 182, 219
Singapore 184, 221
Slovenia 164, 201
South Africa 189, 226
South Korea 173, 210
Spain 164, 201
Sweden 183, 220
Switzerland 158, 195
Taiwan 187, 224
Thailand 185, 222
Turkey 186, 223
United Arab Emirates 153, 190
United Kingdom 165, 202
United States 188, 225

N
National Language Support
  See NLS (National Language Support)
National Language Support (NLS)
  formats 55
  NENGO format 104
  NENGO informat 421
  Netherlands
    monetary format 164, 201
  New Zealand
    monetary format 180, 217
  NLBEST format 105
  NLDATE format 107
  NLDATE function 335
  NLDATE informat 422
  NLDATEL format 108
  NLDATEM format 109
  NLDATEMD format 110
  NLDATEMDL Format 111
  NLDATEMDM format 111, 134
  NLDATEMDS format 112
  NLDATEMN format 113
  NLDATEMYL format 119
  NLDATES format 114
  NLDATEW format 115
  NLDATEW informat 423
  NLDATEWN format 116
  NLDATEYM format 118
  NLDATEYMM format 120
  NLDATEYMS format 121
  NLDATEYQ format 122
  NLDATEYQL format 123
  NLDATEYQM format 124
  NLDATEYQS format 125
  NLDATEYR format 126
  NLDATEYW format 127
  NLDATM format 128
  NLDATM informat 424
  NLDATMAP format 129
  NLDATMAP informat 425
  NLDATMDT format 130
  NLDATML format 131
  NLDATMM format 132
  NLDATMMDF format 132
  NLDATMDMDL format 133
  NLDATMDMS format 135
  NLDATMN format 136
  NLDATMS format 137
  NLDATMTM format 138
  NLDATMTZ format
datetime values 139
  NLDATMW format 140
  NLDATMW informat 426
  NLDATMWN format 141
  NLDATMWZ format
datetime values 142
  NLDATMY format 143
  NLDATMYML format 144
  NLDATMYMM format 145
  NLDATMYMS format 145
  NLDATMYQ format 146
  NLDATMYQL format 147
  NLDATMYQM format 148
  NLDATMYQS format 149
  NLDATMYR format 150
  NLDATMYW format 151
  NLDATMZ format
datetime values 152
  NLDATSEPARATOR system option 556
  NLMNIAED format 153
  NLMNIAED informat 427
  NLMNIAUD format 154
  NLMNIAUD informat 428
  NLMNIBGN format 155
  NLMNIBGN informat 429
  NLMNICAD format 157
  NLMNICADw.d informat 430
  NLMNICADw.d format 157
  NLMNICADw.d informat 431
  NLMNICAF format 158
  NLMNICAFw.d format 432
  NLMNICCN format 159
  NLMNICCNw.d format 433
  NLMNICZ format 160
  NLMNICZK format 434
  NLMNICZKw.d format 434
O
ODS output
  changing language of the text  557
ODSCHARSET option  587
ODSLANGCHG system option  557
OPT= option, TRANTAB statement  592
output processing  17
OUTREP data set option  52
overriding encoding  49

P
paper size and measurement  6
PAPERSIZE= system option
default values  606
Poland
  monetary format  181, 218
Portugal
  monetary format  164, 201
Puerto Rico
  monetary format  188, 225

R
region codes
current two-letter code  318
remote applications
  illegal data  558
right-alignment
  character expressions  386
RSASIOTRANSERROR system option  558
Russia
  monetary format  182, 219

S
SAS data sets
  naming  561
SAS data views
  naming  561
SAS language elements
  using encoding values  613
SAS sessions
  default character-set encoding  549
  encoding behavior in  17
  locale of  550
  specifying locale during  7
SAS string functions
  internationalization compatibility for  281
SASMSIG function  342
SASMSGL function  345
SBCS encoding
  encoding values for transcoding data  613
searching
for a word, by position in a string 536
for specific characters in character expression 381
for string of characters in character expression 380

segments
storing long macro variable values in 539
SETLOCALE function 350
shift out/shift in (SO/SI)
  trimming from character expressions 392

Singapore
  monetary format 184, 221
single-byte character set (SBCS)
  encoding values for transcoding data 613

Slovenia
  monetary format 164, 201
SO/SI (shift out/shift in)
  trimming from character expressions 392

sort keys
  linguistic 347
SORT option 571
SORT procedure
  language-specific collating sequence for 559
SORTKEY function 347
SORTSEQ= system option 559

South Africa
  monetary format 189, 226

South Korea
  monetary format 173, 210

Spain
  monetary format 164, 201

SQL procedure
  language-specific collating sequence for 559

standards organizations 13

string functions
  internationalization compatibility for 281

strings
  length of 535
  locating first character in 534
  reducing length of 536
  searching for words by position in 536
  substring of a character string 538

substrings
  extracting from an argument 390
  extracting from an argument, based on byte position 330
  of a character string 538

Sweden
  monetary format 183, 220

Switzerland
  monetary format 158, 195
  system options
    for transcoding 20

T
Taiwan
  monetary format 187, 224
Taiwanese dates 102, 419
Thailand
  monetary format 185, 222
time 6
time values
  converting to specified locale 340
  Time Zone 25
TIMESZONE= system option 560
  trailing blanks
    trimming 44
  transcode attributes
    of data set variables 372
TRANCODE= Column Modifier on
  PROC SQL 588
TRANCODE= option
  ATTRIB statement 590
transcoding 10, 19
  by specified translation table 367
  compatible and incompatible encodings 21
  considerations for 21
  enabled for specified argument 399
  enabled for specified character variable 398
  encoding values for 613
  line-feed characters 22
  preventing 22
  reasons for 20
  SAS options for 20
  suppressing 51
  transferring data between EBCDIC and
    ASCII 22
  versus encoding 10
transcoding errors 558
translation tables
  applying to transport files 594
  transcoding by specified table 367
transport files
  applying translation tables to 594
TRANTAB function 367
TRANTAB statement
  UPLOAD procedure 592
TRANTAB-ODS option 588, 591
  trimming trailing blanks 44
  truncating numeric values 331
Turkey
monetary format 186, 223
TYPE= option, TRANTAB statement 592
TZ= system option 560
TZID
TZONEID Function 357
TZONEDSTOFF function 363
TZONEOFF function 357
TZONENAME function 359
TZONESTOFF function 360
TZONES2U function 361
TZONESTTNAME function 364
TZONESTTOFF function 365
TZONEU2S function 366

U
Unicode 10
BOM prefix on external files 546
encoding values for transcoding data 613
length of character unit 371
length of display unit 372
Unicode Consortium 13
UNICODE function 368
UNICODEC function 369
UNICODELEN function 371
UNICODEWIDTH function 372
United Arab Emirates
monetary format 153, 190
United Kingdom
monetary format 165, 202
United States
monetary format 188, 225
UPLOAD procedure
TRANTAB statement 592
uploading files
translation tables 592
uppercase characters
changing to lowercase 43
converting arguments to 393
UTF-16 11
UTF-32 11
UTF-8 10

V
VALIDMEMNAME system option 561
VALIDVARNAME= system option 564
variable names
rules for valid names 564
variables
associating formats with 589
associating informats with 589
labels 589
length, associating with 589
transcode attributes of 372
transcoding enabled for specified character variable 398
variant characters 14
VARTRANSCODE function 372
VTRANSCODE function 398
VTRANSCODEX function 399

W
WEEKU format 267
WEEKV format 268
WEEKW format 270
Windows
ISO encodings 11
words
searching for, by position in a string 536

X
XMLENENCODING option 592

Y
YEN format 271
YYWEEKU format 272
YYWEEKV format 274
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.