unicode locale — Unicode locale utilities

Description Syntax Remarks and examples Also see

Description

`unicode locale list` lists all available locales or those locales that meet the specified criteria. Any of these locale codes may be specified in Stata or Mata functions that accept a locale as an argument, such as `ustrcompare()` and `ustrupper()`, or in the `set locale_functions` setting.

`unicode uipackage list` lists all localization packages that are available for the graphics user interface (GUI). Any of the listed locales may be specified in the `set locale_ui` setting to change the language of the text that is displayed in GUI elements such as the menus and dialog boxes.

Syntax

List locales

`unicode locale list` [*pattern*]

List user interface (UI) localization packages

`unicode uipackage list`

pattern is one of `_all`, `*`, `*`*name*`*`, `*`*name*, or *name*`*`. If you specify nothing, `_all`, or `*`, then all results will be listed. `*`*name*`*` lists all results containing *name*; `*`*name* lists all results ending with *name*; and *name*`*` lists all results starting with *name*.

Remarks and examples

Remarks are presented under the following headings:

Overview

A locale identifies a user community with a certain preference for how their language should be written; see **[U] 12.4.2.4 Locales in Unicode**. A locale can be as general as a certain language (for example, "en" for English) or can be more specific to a country or region (for example, "en_US" for US English or "en_HK" for Hong Kong English. Stata uses International Components for Unicode's (ICU's) locale format. See http://userguide.icu-project.org/locale for full information about ICU. Note that ICU differs from the POSIX locale identifiers used by Linux systems.

Locales use tags to define how specific they are to language variants. An ICU locale may contain up to five subtags in the following order: language, script, country, variant, and keywords. Typically, the language is required and the other tags are optional. In most cases, Stata uses only the language and country tags. For example, "en_US" specifies the language as English and the country as the USA.

Many language-specific operations require the locale to perform their task. This kind of operation is called locale-sensitive. For example, in English, the uppercase letter of the Latin small letter "i" is the Latin capital letter "I". However, in Turkish, the uppercase letter is "İ" with a dot above it (Unicode \u0130); hence, the case mapping is locale-sensitive.

The following functions are locale-sensitive: `ustrupper()`, `ustrlower()`, `ustrtitle()`, `ustrword()`, `ustrwordcount()`, `ustrcompare()`, `ustrcompareex()`, `ustrsortkey()`, and `ustrsortkeyex()`.

Although Stata usually uses only the language and country tags, collation keywords may also be used in functions `ustrcompare()` and `ustrsortkey()` to affect ordering of Unicode strings. The collation keyword affects the string sort order of the locale. For example, "pinyin" and "stroke" for Chinese language produce different string sort orders. In most cases, it is not necessary to specify a collation keyword; the default collator (either for Stata or for the language) provides sufficient control. However, some programmers may wish to specify a specific value. If you do not know the value of the collation keyword, you can obtain a list of valid collation values and their meanings in XML format at http://unicode.org/repos/cldr/trunk/common/bcp47/collation.xml.

Default locale and locale fallback

Because a locale is simply an identifier to locate the resources for specific services, there is no validation of the locale. For example, specifying "klingon" is as valid as specifying "en" when calling `ustrcompare()` or the other functions discussed here. If the collation data for the "klingon" locale is found, then the locale is populated; otherwise, a fallback search process starts.

The fallback process proceeds as follows:

1 . The variant is removed if there is one.

2 . The country is removed if there is one.

3 . The script is removed if there is one.

4 . Steps 1–3 are repeated on the default locale.

5 . If a locale cannot be found after following the previous steps, the ICU "Root", or built-in fallback, locale is used.

The process stops at any point if the desired information is found. The ICU default locale is usually the system locale on the machine, which you can change. Note that on macOS, the ICU default locale is usually "en_US_posix", which does not change even if you change the system locale from the operating system's "Language" setting. To see the ICU default locale, you can type

```
. display c(locale_icudflt)
```

You can also find it under the `Unicode settings` in the output of `creturn list` along with two other locale-related settings: `locale_ui` and `locale_functions`. See [P] **set locale_ui** and [P] **set locale_functions** for details.

`set locale_functions` affects the functions `ustrupper()`, `ustrlower()`, `ustrtitle()`, `ustrword()`, `ustrwordcount()`, `ustrcompare()`, `ustrcompareex()`, `ustrsortkey()`, and `ustrsortkeyex()` when no locale is specified. If `locale_functions` is not set, the default ICU locale `c(locale_icudflt)` is used.

For example, if your operating system is Microsoft Windows English version, the system locale is most likely "en". It is "en_US" if you chose the country to be USA during installation of the operating system. If `locale_functions` is not set or is set to `default`, then `ustrupper("istanbul")` is equivalent to `ustrupper("istanbul", "en_US")`, which returns `ISTANBUL`.

However, if `locale_functions` is set to `tr` for Turkish, then `ustrupper("istanbul")` is equivalent to `ustrupper("istanbul", "tr")`, which returns `ISTANBUL` with a dot over the capital I. Although ICU does not validate locales, Stata validates that the language subtag of the `locale_functions` setting is a valid ISO-639-2 language code. (See the ISO-639-2 list at http://www.loc.gov/standards/iso639-2/.) Hence, `set locale_functions klingon` will produce an error.

With the fallback rules, the effective locale can be very different from the locale you specified, depending on the operation being performed. Currently, `ustrword()` and `ustrwordcount()`, which use ICU's word break iterator service, and `ustrcompare()`, `ustrcompareex()`, `ustrsortkey()`, and `ustrsortkeyex()`, which use ICU's collation service, are affected by this. You may use the functions `wordbreaklocale()` and `collatorlocale()` to find the effective locale from the requested locale.

Also see

[D] **unicode** — Unicode utilities

[P] **set locale_functions** — Specify default locale for functions

[P] **set locale_ui** — Specify a localization package for the user interface

[U] 12.4.2 Handling Unicode strings

[U] 12.4.2.4 Locales in Unicode