C++11 Strings and Character Encoding

Previous Next

On the wire, Ice transmits all strings as Unicode strings in UTF-8 encoding. For languages other than C++, Ice uses strings in their language-native Unicode representation and converts automatically to and from UTF-8 for transmission, so applications can transparently use characters from non-English alphabets.

However, for C++, how strings are represented inside a process depends on the platform as well as the mapping that is chosen for a particular string: the default mapping to `std::string`, or the alternative mapping to `std::wstring`.

> This discussion is only relevant for C++. For scripting language mappings based on Ice for C++, it is possible to use Ice's default string converter plug-in and to install your own string converter plug-in.

We will explore how strings are encoded by the Ice for C++ run time, and how you can achieve automatic conversion of strings in their native representation to and from UTF-8. For an example of using string converters in C++, refer to the sample program provided in the `demo/Ice/converter` subdirectory of your Ice distribution.

By default, the Ice run time encodes strings as follows:

- Narrow strings (that is, strings mapped to `std::string`) are presented to the application in UTF-8 encoding and, similarly, the application is expected to provide narrow strings in UTF-8 encoding to the Ice run time for transmission.

  With this default behavior, the application code is responsible for converting between the native codeset for 8-bit characters and UTF-8. For example, if the native codeset is ISO Latin-1, the application is responsible for converting between UTF-8 and narrow (8-bit) characters in ISO Latin-1 encoding.

  Also note that the default behavior does not require the application to do anything if it only uses characters in the ASCII range. (This is because a string containing only characters in the (7-bit) ASCII range is also a valid UTF-8 string.)

- Wide strings (that is, strings mapped to `std::wstring`) are automatically encoded as Unicode by the Ice run time as appropriate for the platform. For example, for Windows, the Ice run time converts between UTF-8 and UTF-16 in little-endian representation whereas, for Linux, the Ice run time converts between UTF-8 and UTF-32 in the endian-ness appropriate for the host CPU.

  With this default behavior, wide strings are transparently converted between their on-the-wire representation and their native C++ representation as appropriate, so application code need not do anything special. (The exception is if an application uses a non-Unicode encoding, such as Shift-JIS, as its native `wstring` codeset.)

Topics

- Installing String Converters with C++11
- UTF-8 Conversion with C++11
- String Parameters in Local C++11 Calls
- Built-in String Converters in C++11
- C++11 String Conversion Convenience Functions
- The C++11 iconv String Converter
- The C++11 Ice String Converter Plug-in
- Custom String Converter Plug-ins with C++11

Back to Top ^

See Also

- The Ice Protocol
- C++11 Mapping for Built-In Types

Previous Next