Doc No: X3J16/96-0135
WG21/N0953
Date: July 8, 1996
Reply to: Aangelika Langer
Email: langer@roguewave.com

Questions and Open Issues Regarding Locale

Message Facets

The message facet has a member function open() for opening a message catalogue. This function has to be provided with a reference to a locale on each call. This is because the message facet needs a codecvt facet for converting the messages retrieved from a catalogue for further processing inside the program.

Question: Is there any need to associate a different locale to each message catalogue?

Assumed answer: No. Most likely one would expect that the message facet uses its "sibling" code conversion facet in all cases, i.e. the code conversion facet that is contained in the same locale.

Problems:
1. The interface as it is right now is error-prone and counter-intuitive. The message facet's open() function requires a redundant reference to a locale. (At least, if my assumption is right, the reference is redundant.) Redundancies are in general error-prone. As a user I can inadvertently provide the wrong locale, i.e. one that is different from the locale that contains the message facet I use.

2. What does it semantically mean that a different locale is associated to each message catalogue that gets opened by a certain message facet? The associated locales contain message facets themselves, not only the code conversion facet that is supposed to be used in subsequent message text retrievals from that catalogue. This is a confusing concept.

3. There is no uniform mechanism for making required facets available to another facet. The message facet needs a code conversion facet, which is provided as a reference to a locale to the open() function. The numeric facets, for instance, depend on the numpunct facet. Here the required facet is provided via a reference to an ios_base object that contains a locale. The put() and get() function originally had a locale parameter, like open(). This parameter was later dropped because of its redundancy. Now, the mechanisms for providing dependent facets are different and even more difficult to explain and more confusing.

Conceivable Resolutions:
1. Provide the open() call with a reference to a code conversion facet, instead of a reference to an entire locale. This would mitigate the confusion about the semantic meaning of a locale that is associated to a message catalogue. This is not too good an idea though, because facets are not supposed to be used stand-alone; for instance their destructor is private, i.e. instances of facets are not supposed to be created independently of a locale. Plus, this solution does not provide a uniform approach to describe interdependencies among facets in a locale.

2. Make locales "self-contained" in the sense that interdependencies among facets are generally resolved among "sibling" facets, i.e. a facet that needs other facets will use those that are contained in the same locale. This requires to provide facets with the knowledge about their containing locale or, alternatively, with the knowledge about its "sibling" facets.

I don’t have any proposed solution to this issue right now. However, I would like to hear from the working group whether there is any compelling reason against "self-contained" locales, which is the semantics many users will expect when thinking of a locale.

Stream Operators for Class Locale

The shift operators for class locale insert a locale name including a subsequent endl manipulator. Consequently the extractor expects a locale name to fill an entire line. This deems nonsensical to me, especially as the endl manipulator flushes the stream, which is time consuming and most likely not intended.

Proposed Resolution: Drop the requirement of inserting the endl manipulator and the requirement of extracting a locale name from a entire line.

Time Parsing

The time_put facet allows to specify format strings like the ones for the C function strftime(). However, the time_get facet does not allow to parse date or time units according to specifications used by strptime(), which to my knowledge is the counterpart to strftime() in X/Open.

Problem: To my mind this is an omission from the draft, which in other cases tries to provide services equivalent to what is available in the C library. POSIX does not specify a strptime() function, but the C++ locale seems to aim for X/Open coverage. See the message facet, for instance, which is not part of a C locale, but contained in X/Open locales and in the C++ locale.

Resolution: Add the missing functionality to the time_get facet.

I don’t have any proposed solution to this issue right now. However, I would like to hear from the working group whether this omission is intended and why.

Numeric Parsing

. Box 64 says that the numeric parsing facet needs a locale, but it is not clear where that locale is coming from. Why isn’t it taken from the ios_base parameter that is provided to each call to the get member function?
. Some garbled sentences in section 22.2.2.1.2.

Numeric Formatting

The description of stage 4 refers to a failed() member function of the output iterator. Is there a partial specialization required for pointers, which are output iterators and do not have a failed() member function?