Remove Deprecated Locale Category Facets For Unicode from C++26

Document #:	P2873R0
Date:	2023-05-15
Project:	Programming Language C++
Audience:	SG16 Unicode
Revises:	N/A

Reply-to:	Alisdair Meredith <ameredith1@bloomberg.net>

Contents

1 Abstract

Several locale facets were added to C++11 that were intended to support Unicode transcoding. These facets have been deprecated since C++20 per the recommendation of our Unicode Study Group, SG16, due to poor error handling capabilities when handling potentially malformed Unicode byte sequences. This paper proposes removing those facets from the C++ Standard Library.

2 Revision history

2.1 R0: Varna 2023

Initial draft of the paper.

3 Introduction

At the start of the C++23 cycle, [P2139R2] tried to review each deprecated feature of C++ to see which we would benefit from actively removing and which might now be better undeprecated. Consolidating all this analysis into one place was intended to ease the (L)EWG review process but in return gave the author so much feedback that the next revision of the paper was not completed.

For the C++26 cycle, a much shorter paper, [P2863R0], will track the overall analysis, but for features that the author wants to actively progress, a distinct paper will decouple progress from the larger paper so that the delays on a single feature do not hold up progress on all.

This paper takes up the deprecated locale category facets for Unicode, D.28 [depr.locale.category].

4 History

This feature was added as part of the initial basic support for Unicode types in C++11 by paper [N2238] and deprecated on the recommendation of SG16 for C++20 by paper [P0482R6].

5 Feedback

5.1 Initial C++23 Review: telecon 2020/07/13

Discussion was broadly in favor of removal from the C++23 specification, and relying on library vendors to maintain source compatibility as long as needed. However, LEWGI explicitly requested to confer with SG16 in case that study group is aware of any reason to hold back on removal, before proceeding with the recommendation.

5.2 SG16 Review: telecon 2020/07/22

SG16 is concerned that `code_cvt` in general has poor error handling facilities, especially when dealing with encodings that may take multiple code units to express a code point, and so have more cause to report on malformed inputs. The specific facets in D.28 [depr.locale.category] stand in the way of putting a minimally useful replacement into the standard, by sitting on the good names but with poor semantics. This means that the usual safety net of the zombie names clause does not apply, as we will want code to not compile for at least one release in order to introduce a replacement with the same names but more appropriate semantics. Concern was raised about removing a feature deprecated only as recently as the current C++20 Standard.

Polling showed no consensus to recommend the removal for C++23, but no objection to that removal either.

5.3 LEWGI Consensus:

Confirmed SG16 has no objection; remove this feature from C++23

6 Proposal

Remove Deprecated Locale Category Facets For Unicode from C++26.

7 Wording

All wording is relative to [N4944], the latest working draft at the time of writing.

D.28 [depr.locale.category] Deprecated locale category facets The `ctype` locale category includes the following facets as if they were specified in table Table 104 of 30.3.1.2.1 [locale.category].

```
    codecvt<char16_t, char, mbstate_t>`
    codecvt<char32_t, char, mbstate_t>
```

[2] The ctype locale category includes the following facets as if they were specified in table Table 105 of 30.3.1.2.1 [locale.category].

```
codecvt_byname<char16_t, char, mbstate_t>
codecvt_byname<char32_t, char, mbstate_t>
```

[3] The following class template specializations are required in addition to those specified in 30.4.2.5 [locale.codecvt]. The specialization `codecvt<char16_t, char, mbstate_t>` converts between the UTF-16 and UTF-8 encoding forms, and the specialization `codecvt<char32_t, char, mbstate_t>` converts between the UTF-32 and UTF-8 encoding forms.

8 Acknowledgements

Thanks to Michael Parks for the pandoc-based framework used to transform this document's source from Markdown.

9 References

[N2238] Matthew Austern. 2007-04-17. Minimal Unicode support for the standard library (revision 3).
https://wg21.link/n2238

[N4944] Thomas Köppe. 2023-03-22. Working Draft, Standard for Programming Language C++.
https://wg21.link/n4944

[P0482R6] Tom Honermann. 2018-11-09. char8_t: A type for UTF-8 characters and strings (Revision 6).
https://wg21.link/p0482r6

[P2139R2] Alisdair Meredith. 2020-07-15. Reviewing Deprecated Facilities of C++20 for C++23.
https://wg21.link/p2139r2