# Remove `wstring_convert` From C++26


| | |
|---|---|
| Document #: | P2872R0 |
| Date: | 2023-05-15 |
| Project: | Programming Language C++ |
| Audience: | Library Evolution Incubator |
| Revises: | N/A |


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


## Contents



## 1 Abstract


The `wstring_convert` library component has been deprecated since C++17. As noted at the time, the feature is underspecified and would require more work than we wish to invest to bring it up to standard. This paper proposes removing that facility 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 C-style API for race-free access to `shared_ptr` objects, D.24 [depr.util.smartptr.shared.atomic].

# 4 History

This feature was originally proposed for C++11 by paper [N2401] and deprecated for C++17 by paper [P0618R0]. As noted at the time, the feature was underspecified and would require more work than we wished to invest to bring it up to standard. Since then SG16 has been convened and is producing a steady stream of work to bring reliable well-specified Unicode support to C++.

Given vendors propensity to provide ongoing support for these names under the zombie name reservations, the strong recommendation of this paper is to remove this library immediately from the C++23 Standard. The weak recommendation is to do the minimal work to clean up the wording to use the more precise terms that replaced Requires clauses, waiting until SG16 (or some other entity) produces a clean replacement for this facility for users to migrate to before removal.

There are currently 4 open LWG issues relating to this clause; there would be more, but we would rather see this feature removed than keep adding issues to deprecated library features.

— [LWG2478] Unclear how `wstring_convert` uses `cvtstate`
— [LWG2479] Unclear how `wbuffer_convert` uses `cvtstate`
— [LWG2480] Error handling of `wbuffer_convert` unclear
— [LWG2481] `wstring_convert` should be more precise regarding “byte-error string” etc.

# 5 Feedback

## 5.1 Initial Review for C++23: 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 for C++23: telecon 2020/07/22

SG16 raises concerns that the original paper deprecating this feature lacked a strong motivation, as it was simply recording a recommendation from LWG review when deprecating the `<codecvt>` header for D.20. There is general concern that `codecvt` is not fit for purpose, notably due to poorly specified error handling capabilities while transcoding, but this deprecation does not address that, it is merely a convenience API for using that underspecified library component. While removing the `<codecvt>` header might mean there would be fewer standard `codecvt` facets to use with this API, it remains just a as usable with user provided `codecvt` facets as before, as well as those in the `<locale>` header. While we would like to see a replacement facility, there is no such proposal at this time.

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

## 5.3 LEWGI Consensus for C++23:

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

# 6 Recommendation for C++26

Following LEWGI consensus for C++23, recommend the removal of this

Close LWG issues [LWG2478], [LWG2479], [LWG2480], and [LWG2481] as resolved by removal of the feature in paper [P2863R0].

# 7 Acknowledgements

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

# 8 References

[LWG2478] Jonathan Wakely. Unclear how wstring_convert uses cvtstate.
https://wg21.link/lwg2478

[LWG2479] Jonathan Wakely. Unclear how wbuffer_convert uses cvtstate.
https://wg21.link/lwg2479

[LWG2480] Jonathan Wakely. Error handling of wbuffer_convert unclear.
https://wg21.link/lwg2480

[LWG2481] Jonathan Wakely. wstring_convert should be more precise regarding "byte-error string" etc.
https://wg21.link/lwg2481

[N2401] P.J. Plauger. 2007-09-03. Code Conversion Facets for the Standard C++ Library.
https://wg21.link/n2401

[P0618R0] Alisdair Meredith. 2017-03-02. Deprecating <codecvt>.
https://wg21.link/p0618r0

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

[P2863R0] Alisdair Meredith. 2023-05-15. Remove `wstring_convert` From C++26.
https://wg21.link/p2872r0