# Initial draft proposal for core language UB white paper: Process and major work items


| | |
|---|---|
| Doc # | P3656 R1 |
| Authors | Herb Sutter, Gašper Ažman |
| Date | 2025-03-23 |
| Audience | EWG |


## Abstract: Background and scope


At Hagenberg 2025-02, EWG encouraged the following work:

> Poll: Pursue a language safety white paper in the C++26 timeframe containing systematic treatment of core language Undefined Behavior in C++, covering Erroneous Behavior, Profiles, and Contracts. Appoint Herb and Gašper as editors.
>
> | SF | F | N | A | SA |
> |---|---|---|---|---|
> | 32 | 31 | 6 | 4 | 4 |

Details about white papers are available at SD-4 > Technical specifications and white papers.

The first step is to get EWG input and work to consensus on process and method:

- **Process: How we intend to work and gain consensus.** This paper proposes EWG approval (possibly in telecon) for each item to be adopted into the white paper working draft. EWG might direct some specific discussion happen in an SG first, for example to see a proposed addition in SG23 (Safety and Security) before bringing it to EWG.
- **Major work items: What things will need to be done.** This paper proposes four main work items to enumerate and address core language UB.


## Changes since R0

Implemented EWG 2025-03-19 telecon feedback: Scope includes IF-NDR. EWG reviews published P papers, not individual GitHub PRs. The starter set of options for dealing with UB is not exhaustive, and things like nondeterministic rules would require papers.

# 1. Proposed process

This paper proposes that we:

1. start with an empty white paper;
2. iteratively get EWG approval (possibly in telecon) for each item to be adopted into the white paper working draft (i.e., we suggest EWG initially owns the draft working paper even before plenary assumes ownership); and
3. finally as usual get EWG approval to forward the result to CWG for wording and then plenary for approval to send the white paper to SC 22 for their publication vote.

We suggest that to maintain the draft white paper we:

- Have a public GitHub repo for the editing, like we do with the IS. EWG reviews P papers as usual.
- Use pull requests: Iterations occur on the PR. EWG approves published diff P papers to be merged via these PRs.
- Use issues to track editing: This tracks rationale, can link to supporting papers, etc. These are used by the white paper editors for their own management purposes. They could also be used to summarize discussion on the mailing lists so everyone feels caught-up.
- Use the reflectors for discussion: These track discussion as usual (including on a designated, or new, subgroup/reflector if EWG chairs decide that).

For each meeting:

- Tag a release.
- Publish the current draft white paper as a Pxxxx Rnext paper, with a list of changes from Rprev (export from GitHub) and issues.
- Request WG 21 member review comments, to be submitted as GitHub issues.

Notes:

- For a given proposed addition, EWG might direct the author to get feedback from an SG, for example to get security feedback in SG23 (Safety and Security) and bring the possibly-updated paper together with that feedback also to EWG.
- This section says "EWG" because the mandate for this work originated in EWG, and the mandate scope was a white paper on UB in the *core language* rather than in the standard library. However, if we find areas that touch library, of course we should also include LEWG for those.

# 2. Proposed major work items

This paper proposes four main work items.

## 2.1 List cases: Enumerate language UB (includes IF-NDR)

Goal: End up with a complete list of all language UB. For the rest of this paper, "UB" includes IF-NDR for brevity (as directed in EWG 2025-03-19 telecon poll).

Today, just searching for "undefined" isn't enough to find all UB in the standard and systematically list it. We need to systematically list core language UB throughout the standard.

Thoughts on how:

- Crowdsource: Call for volunteers to sign up for a clause.
- Check: Against other (sometimes partial) sources, notably Shafik Yaghmour's papers P1705 and P3075, P3100 authors' list so far, compilers' constexpr evaluator checks.
- Review: Round of review with CWG experts (or by CWG?) – what did we miss?

At this stage we can do basic categorization, for example:

- Which ones are security-related? Have security experts tag which have security impact always, never, or sometimes.
- Which are efficiently locally diagnosable? Note the type of information needed to diagnose violations, for example whether a compiler can see it locally vs. there is escaped information that requires extra state and sanitizer-like tools.

Specific suggestion: Add a LaTeX \tag{} for UB right in the working draft sources (spelling TBD, ask edit reflector experts).

- Since adding LaTeX tags can be done without disturbing the generated standard, they are editorial and we can start adding them via PRs to the IS working draft itself.
- This allows them to be maintained together with the IS and minimize getting out of sync.
- This also allows easily generating a non-normative UB Annex in the standard (if we want).

In the white paper, repeat the list of UB and add more non-editorial information (e.g., a description and example as in Shafik's paper, safety/security considerations, and similar notes).

## 2.2 List tools: Create "non-exhaustive starter menu of tools"

Goal: Get agreement on the basic options we can choose from to deal with a given case of UB. This section is to become a separate P paper that proposes actual wording to add to the draft white paper, if approved by EWG.

For each case of UB, the key question is "what do we do about it?" We want to generate a list of possible options we can pick from to decide how to handle each case of UB.

The Hagenberg 2025-02 EWG poll already suggested three possible basic tools:

- Erroneous Behavior (already in C++26)
- Profiles (not in C++26, white paper will need to add or reference, e.g., P3589R2)
- Contracts (already in C++26, except labels/grouping which white paper will need to add or reference, e.g., from P3400 (see 2.4))

So a candidate list might be:

- Make the UB well-defined (possibly a check)
- Make the UB ill-formed
- Make the UB rejected (does not affect SFINAE), always or when a profile is enforced
- Make the UB deprecated, and possibly also ill-formed or rejected when a profile is enforced
- Make the UB instead be EB in the language (which includes defining a semantic, possibly a contract_assert check), either always (on by default, like uninitialized locals in C++26) or via opt-in (to be grouped into actual profiles/labels later)
- With a prominent note that these are not exhaustive: Additional ideas are welcome: Papers please!

The initial core set would not include the following; these would need papers that explain why EWG should consider them in specific cases:

- nondeterministic heuristics
- hardware support

Finally, if possible we should determine guidelines for when to use each tool, particularly what are the considerations for each:

- performance considerations
- adoption hurdles like crashes
- etc.

## 2.3 Apply: Take a first pass at penciling in which tool to use for each UB case

Papers please! This requires design.

Could do it one clause at a time

- Crawl-walk-run: Get feet wet with one clause first.
- Start with some interesting clause (e.g., wherever array subscript out-of-bounds UB would be covered).

Each UB case could be a GitHub issue with its dedicated discussion thread

- this can link to your papers
- maintains a record of the discussion of this EB case to document rationale
- also separates the discussion of this EB case from the others (hopefully more manageable than a single email list that has a huge volume of hard-to-sort traffic)
- lets people vote on individual comments/answers

### 2.4 Group: Group UB cases (profile names / contract labels)

Identify cohesive groups of UB that want to be addressed together

- "Establish cohesion in solution spaces."
- Can overlap: A given case could be under more than one group.

Ideally crowdsource initial suggestions?

- Make a list of possible labels/profiles.
- Do another spreadsheet "voting table" for groupings.

## Status and what's next

The EWG 2025-03-19 telecon strongly encouraged following the direction in this paper

> POLL: P3656R0: EWG believes P3656 is ‘on the right track’ with the strategy proposed for producing a white-paper for “Core Language UB (and IF-NDR)”
>
> | SF | F | N | A | SA |
> |---|---|---|---|---|
> | 8 | 13 | 1 | 0 | 0 |

The next steps will include creating a GitHub repo for an initially empty draft white paper, and generating content via P papers to be approved by EWG to add to the draft white paper.

# Appendix: Suggested guidance for tags and descriptions

> The following helpful guidance suggestion came in after the EWG 2025-03-19 telecon, and it seems useful so we want to capture it in this revision. However, we want to be clear that it was **not yet reviewed** in an EWG telecon, so we're putting it in this distinct section.

Here is a suggested checklist of some information we expect to gather about each instance of UB, in the IS source tag (including the generated IS Annex) and in the separate white paper. We want to capture all of the information described in P3075, either in the IS (tag and Annex) or the white paper.

In the IS source, the purpose of the UB tag is to mark the basic existence of UB.

- If the UB is explicit, we should tag the specific phrase that introduces it. As described in P3075, explicit UB includes looking for these phrases: "the behavior of the program is undefined," "has undefined behavior," "results in undefined behavior," "the behavior is undefined," "have undefined behavior," "is undefined," "result has undefined behavior;" and explicit IF-NDR includes looking for these phrases: "no diagnostic is required," "no diagnostic required," and "no diagnostic shall be issued."
- Otherwise, we just mark that there is UB. In the white paper, describe how it arises. In the Core issues list, open a core issue to make the UB explicit (and refer to it in the white paper).

In the white paper, we have room for more detailed discussion about each case of UB. It should include:

- A title giving a concise summary of the UB.
- The subclause with a numeric cross-reference to the main IS text.
- A short summary of the UB. Use "can" and "could," avoid normative "shall," "may," "might."
- An example code that demonstrates the UB. Use "// error" in code comments.
- Categorization of the UB regarding whether it can be identified with a simple check based on information already available in the grammar or the abstract machine, or if it needs additional instrumentation of some kind.
- List of potential mechanisms to address the UB.
- Wording for the proposed solution (if there is one being proposed).

While we are doing the initial pass to gather the list of UB, some of the above can be initially marked "TODO", just so we don't forget to go back and include it.

Having a common template for the information to be gathered will keep everything consistent. From this information, we could then produce a few things based on what we decide to pursue

- An index of UB using just the short descriptions
- An annex listing more details of the UB that could be added to the standard immediately
- The contents of the white paper's wording section proposing actual normative changes to address the UB