Severing P3100's Profiles Claim from Its Case-by-Case Review

Document Number:	P4297R0
Date:	2026-07-14
Intent:	Ask
Audience:	EWG

Reply-to:	Vinnie Falco vinnie.falco@gmail.com Ville Voutilainen ville.voutilainen@gmail.com

Table of Contents

Abstract

This paper asks EWG (the Evolution Working Group) to sever an unadopted architecture claim from the wording it is bundled with, so that the wording proceeds and the claim gets its own paper and poll.

P3100R8 bundles two things that can be judged separately: proposed wording for 77 runtime-checkable cases of undefined behavior, and an architecture claim that Profiles are a higher-level feature built on top of P3100's machinery. The claim decides who owns the configuration of runtime checking. The wording is now under case-by-case review in EWG; its first six

foundational clauses carry the architecture, and once they pass, each remaining case is a mechanical application that gains standing for a claim no poll has ever decided.

This paper reconstructs the proposal's seven-poll history and shows none adopted the architecture, reports that no published paper contests the characterization, and illustrates that the layering question is substantive, contested, and grounded in a decade of deployment evidence. It proposes three polls: a scope statement that wording approvals do not adopt the layering, a process commitment that the layering requires a dedicated paper and explicit poll, and an intent statement that EWG will weigh deployment experience when that poll is taken.

Revision History

R0: July 2026

- Initial version. An earlier working draft circulated before publication asked EWG to defer case-by-case wording review pending implementation and deployment experience. This published version withdraws that request. The review proceeds on its merits, and the ask is stated in Section 7 as three polls. This version adds the Brno 2026-06 poll to the poll history (Table 1, row 7, sourced from the public paper tracker) and identifies the foundational wording clauses that carry the architecture (Table 2).

---

1. Introduction

P3100R8 pairs proposed wording for 77 runtime-checkable cases of core-language undefined behavior with an architecture claim: that Profiles are a higher-level feature built on top of that wording's machinery. The wording is headed into case-by-case EWG review. Alongside it, the architecture claim advances without a ballot of its own. This paper asks EWG to sever the two, so wording review proceeds and the layering is decided by a ballot written for it.

The layering claim sits between two competing bodies of published work: P3100R8's implicit contract assertions, configured through the Labels facility of P3400R3, and the Profiles work of P3984R0, P3081R2, and P3589R2. Section 2 sets both out and shows where P3100R8 places Profiles beneath its own tools. Two companion papers in the July 2026 mailing take up adjacent questions: P4306R0 [1] supplies the dedicated comparison of the two configuration-ownership models that this paper's Poll 2 contemplates, and P4310R0 [2] examines the separate question of the response to a detected core-language violation.

This paper contributes three things:

1. It reconstructs the proposal's poll history and classifies what each poll asked (Section 3).
2. It shows that the layering question is substantive, contested, and grounded in a decade of field evidence (Section 4).
3. It reports, by a disclosed and re-runnable method, that no published WG21 paper contests the characterization (Section 5).

The concern behind the ask rests on one assumption, stated plainly so a delegate can test it: a design settled through accumulated approvals, with no single poll deciding it, is harder to revisit than one decided by an explicit ballot. Section 3

gives the evidence for it, and the reader who rejects the assumption can weigh the polls in Table 1 directly.

---

2. P3100 Characterizes Profiles as a Preset over Its Machinery

Section 1 named the two competing bodies of work. On one side, P3100R8's implicit contract assertions - checks the language inserts at each point of undefined behavior - configured through Labels, the in-source facility of P3400R3 [3]. On the other, the Profiles work: named, enforceable safety guarantees with the Profiles framework as the feature the user configures directly. They can coexist, but they cannot both be the foundation. Exactly one owns the guarantees and the response to a failed check, and the other is defined in its terms:

- **P3100-first.** P3100's machinery is the foundation. A profile is a named preset that selects from that machinery's settings.
- **Profiles-first.** The Profiles framework is the foundation. It owns the guarantees and the response to a failed check, and P3100's tools sit underneath it.

P3100R8's Appendix A enumeration of every case of explicit core language undefined behavior (80 cases, classified by diagnosability), its five named evaluation semantics that give existing vendor mechanisms a coherent vocabulary, and its backward-compatible wording that invalidates no existing implementation are real achievements independent of the layering question. What follows is about the architecture claim alone.

P3100R8 characterizes Profiles as a preset layered on top of its own tools. Its Section 4.4 states the characterization in conditional terms, but the paper does not rest on the hedge: as the Figure 4 caption below shows, the same layering is stated declaratively, and Section 5.6 (discussed later) acts on it. Section 4.4, under "Configuration":

> *It therefore seems logical to define Profiles as a higher-level feature building on top of these three basic tools (see Figure 4). Given that these three features are configurable, a concrete profile could be defined as being a named configuration preset for these features.*

Its Figure 4 draws the same layering, and the caption states it:

> *Figure 4: Overview of the proposed holistic strategy for removing UB from the C++ language: seven orthogonal tools plus Profiles as a higher-level feature specified on top of these tools. The red rectangle in the centre illustrates the scope of the proposal in Sections 5 and 6 of this paper.*

Its Section 7.2 states what the layering means for P3589R2. Granular control of the evaluation semantics must belong to exactly one feature:

> *For granular, in-source control of the evaluation semantics of implicit contract assertions, we need to agree whether this happens via directives such as the ones proposed in [P3400R3] and shown here, or by using the syntax proposed in the Profiles framework as proposed in [P3589R2]. If we want to have both, we need to specify one in terms of the other to avoid an incoherent and messy design.*

And the same section offers Profiles two futures: a profile "can be defined as essentially a declaration that expands to [P3400R3] directives", or Profiles can be redesigned "as an auditing feature rather than a configuration feature", where a profile no longer configures anything and instead renders a program ill-formed when configuration chosen elsewhere violates its guarantees [4].

Across its own revisions, the proposal's characterization of Profiles has not been stable. P3100R2 [5] (May 2025) stated the opposite position: a profile "should never dictate whether a runtime check is enabled or disabled or what should happen if that check fails". P3100R4 [6] (August 2025) and every revision since states the current one: a profile "could be defined as being a named configuration preset for these features". Same paper number, opposite claim. The reversal itself was never polled. The one adjacent ballot, the Sofia endorsement of slide 53 as "a good basis" (June 2025, Table 1 poll 4, discussed in Section 3), fell between the two revisions and endorsed the slide that carries the new characterization.

The proposal has also acted on the claim once. In its Section 5.6, P3100R8 withdraws the `detection_mode` enumerators from its own proposed wording and characterizes the relationship to P3081R1 [7] this way:

> *... unlike earlier revisions of this paper and unlike [P3081R1], which adopted its library API from those earlier revisions, we no longer propose to add new enumerators to the enumeration detection_mode to encode the category of error (Initialization, Bounds, and so on); instead, this encoding can be accomplished more effectively and flexibly via Labels (see Section 7.1).*

The consequence is concrete. P3081R2's proposed wording still adds those enumerators - `detection_mode::type`, `detection_mode::bounds`, and `detection_mode::lifetime`, each defined as indicating that "the contract assertion was evaluated as part of" the corresponding profile, with violation handling passing "the detection_mode value corresponding to P" - while P3100R8 proposes none. Those checks were to be delivered by P3100's machinery, as the P3100 authors' own P3543R0 states (Section 5). When P3100R8 moved category encoding to Labels, P3081R2's enumerators lost the mechanism that would produce them. P3081R2, from February 2025, remains the latest revision [8], and no revision has answered. When one feature owns the configuration mechanism, a change to it can leave another paper's wording stranded through independent revision, without the coordination that P3100R8's own Section 7.2 now says is needed.

Read together, P3100R8's Section 4.4, Figure 4, Section 5.6, and Section 7.2 make one claim: P3100's machinery is the base, and Profiles are either a preset that configures it or an auditor that checks it. Either way, P3100 is the foundation and Profiles are defined in its terms.

P3100R8 therefore contains two things that can be evaluated separately.

The first is wording: 77 runtime-checkable cases of undefined behavior, each with a proposed transformation, each standing on its own technical merits regardless of whether Profiles sit above or below P3100's machinery.

The second is the architecture claim, which depends on the wording and cannot advance without it. It is nonetheless separable: it advances alongside the wording review and gains standing from each approval, yet the review never directly examines or polls it.

In short, reviewable wording travels with an architecture claim that has never been polled on its own. Without explicit input from EWG, approving the outcomes of the case-by-case review may close the evolution path for Profiles. The ask is severance - let the wording proceed on its merits, and require the architecture claim to be decided by its own paper and poll.

---

3. Seven Polls Advanced the Paper; None Adopted the Architecture

We examine the proposal's poll record: first why every poll about the proposal looks low-stakes, then what the polls actually decided.

Because the wording requires nothing of any implementation, the polls look low-stakes. P3100R8's Section 5.2:

> *Note that no implementation is actually required to implement these checks: a valid implementation choice is to make all 77 cases always have the ignore semantic. It follows that all existing implementations of C++ are already conforming with this wording transformation.*

Every evaluation semantic is implementation-defined per case, and the proposal's own wording states that "There is no requirement that any particular semantic choice be available for the implicit contract assertion" [4]. This design has a legitimate engineering rationale - the wording is adoptable without breaking any implementation - and a procedural effect: because a poll about it compels no vendor and breaks no code, each vote is easy to cast and easy to justify. But requiring nothing is not the same as deciding nothing. The votes still accumulate toward something concrete: the layering in P3100R8's Figure 4 and the configuration ownership in its Section 7.2.

P3100R8's Section 2, "History and polls", lists what it presents as the committee history through Croydon (March 2026). At the time of writing, P3100R8 itself, dated July 2026, is the latest revision. Rows 1-6 of Table 1 reproduce every poll from that section, quoted from the paper, with the tallies the paper itself gives. There is no independent public list to check those six against, except as noted below. Row 7 is a later poll, taken at Brno in June 2026, after the period Section 2's history covers. It is absent from P3100R8's self-reported history and is quoted here from the public WG21 paper tracker.

Table 1: The six polls in P3100R8's self-reported history (its Section 2, rows 1-6), plus a seventh poll taken at Brno after that history's coverage (row 7, from the public paper tracker). Poll text abridged only by ellipsis. Tallies and result labels as printed in the cited source. The rightmost column classifies what each poll asked for. SG21 is the Contracts study group, SG23 the Safety and Security study group, and EWG the Evolution Working Group.

#	Body, meeting	Poll (as quoted in P3100R8)	SF/F/N/A/SA	Result (as printed)	What was asked
1	SG21, Wrocław, 2024-11	"We support the direction of P3100R1 and encourage the authors to come back with a fully specified proposal."	19/6/0/0/0	Consensus	Direction
2	EWG, Hagenberg, 2025-02	"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."	32/31/6/4/4	Consensus	A white paper and its editors
3	EWG, Sofia, 2025-06	"EWG encourages more work on P3100R2 and wants a step-by-step systematic review of P3100R2 to do per-clause approval for inclusion in the core language UB whitepaper (in telecons)"	15/27/2/0/0	Consensus	A review process
4	EWG, Sofia, 2025-06	"EWG agrees that Timur's (magic) slide 53 in P3754R0 is a good basis for the core language UB whitepaper, and asks that the Whitepaper editors make it so."	14/28/2/1/0	Consensus	A diagram as a basis for the white paper
5	SG23, Kona, 2025-11	"SG23 supports the direction of P3100R4 and recommends its inclusion in C++29"	15/12/1/1/2	Consensus	Direction and a target
6	EWG, Croydon, 2026-03	"Update P3100R5 by applying the presented rules to all cases of runtime-checkable UB in the standard, as listed in appendix A, and bring it back to EWG for case-by-case wording review"	39/19/5/2/1	Strong consensus	A wording update across all 77 cases, and more review
7	EWG, Brno, 2026-06	"EWG Approves of the overall direction of P3100R7, agrees to attend/spend time reviewing every line item in Telecons, and re-consider this in Búzios."	16/15/6/2/0	Consensus	Direction, a telecon commitment, and reconsideration at Búzios

The Hagenberg poll and its tally are independently public in P3656R1 [9], the white paper process document by its appointed editors. On the WG21 paper tracker [10], the Brno poll (row 7) and its tally are independently public. The remaining tallies are

as self-reported in P3100R8.

Three observations follow from the table:

First, none of the seven polls adopts anything. Two are direction polls, one creates a white paper and appoints its editors, one endorses a diagram as a basis for that white paper, one recommends a target, and one requests a wording update and further review. At Brno, the seventh approves the overall direction and commits EWG to line-item telecon review with reconsideration at Búzios. The one poll that established a content-approval process - Sofia's per-clause approval in telecons - never ran. The paper itself reports:

> *However, the telecons for per-clause approval of this paper into the white paper were never scheduled, and the pursuit of the white paper as a ship vehicle has stalled. To make progress, we decided to instead target C++29 with the present proposal.*

Second, the endorsements were for a white paper that no longer exists. The Hagenberg poll created a joint, editor-curated white paper "in the C++26 timeframe", and the Sofia polls endorsed a diagram and a process for that white paper. The white paper stalled, the proposal retargeted to C++29 as an ordinary International Standard track paper, and the endorsements remain in its history section, now attached to a target those polls never named.

Third, the paper's own characterization of this record is stronger than the poll texts and tallies. Its Section 1 states: "The proposed design has been reviewed and approved by SG21, SG23, and EWG." [4] Table 1 records no adoption poll. Within its Section 2, the prose introducing the Kona poll reads "approved it with strong consensus". The poll box directly beneath records "Result: Consensus", with one Against and two Strongly Against, and the poll text asks about direction, which in WG21 procedure does not adopt or approve a design.

The accumulation works in numbered steps.

1. Each recorded consensus becomes the starting point for the next question.
2. Each one raises the cost of objecting later, because a subsequent objection must unseat a standing result rather than address an open one.
3. The Croydon poll routes the proposal into review of 77 cases, one at a time, and each case is a small, technical, reasonable question, none of them the architecture question of P3100R8's Section 4.4.
4. When the last case is approved, the architecture may be settled in effect without ever being polled on its own.

This is a ratchet: a series of small forward steps that are individually easy and collectively hard to undo. Nothing here says the outcome is inevitable. The architecture's one ballot appearance was as "a good basis" for the abandoned white paper (Table 1, poll 4).

The review's structure shows where that architecture lives. P3100R8's wording (its Section 6) is not 77 independent edits: it rests on six foundational changes that establish the framework, after which each individual case is a mechanical application of it. Table 2 lists them.

Table 2: The six foundational wording changes in P3100R8's Section 6 that carry the architecture. Once these are approved, the remaining individual cases are mechanical applications of the framework they establish.

Stable name	Change	What it settles
[defns.undefined]	Redefines undefined behavior as behavior that begins with an implicit contract assertion that the behavior cannot occur	Every case of UB is definitionally a contract assertion, and every later paper writes against this definition
[defns.unconstrained]	Adds a definition for the residual state previously called "undefined behavior" in the specification machinery	Frees the old term to carry its new, contract-based meaning
[intro.abstract] 3+a	Establishes a guarding contract assertion for every operation described as undefined behavior	Attaches the mechanism to every case of UB by construction
[basic.contract.general]	Splits all contract assertions into explicit (user-written) and implicit (compiler-generated)	Makes implicit assertions a first-class language concept rather than a tooling detail
[basic.contract.eval]	Adds the assume semantic and restricts it to implicit assertions	Puts the backwards-compatibility escape hatch and the explicit/implicit asymmetry into normative wording
[basic.contract.implicit]	Adds the section defining implicit contract assertions, stating that all undefined behavior has a guarding contract assertion	The clause that makes the whole framework normative

These six clauses front-load the ratchet: the architecture is decided when they are approved. Once they stand, each remaining case is the small, technical, reasonable question that step (3) describes - change "the behaviour is undefined" to "there is an implicit precondition assertion that this does not occur." A member reviewing the fortieth case is no longer positioned to reopen the framework, because that decision was made when the foundational clauses passed. The ratchet is not 77 equal steps but six foundational ones followed by the rest as mechanical applications.

To the response that direction polls are only encouragement and the real decision comes later: the defaults arrive earlier. By the time an adoption poll exists, the characterization will have seven recorded consensus results and 77 case approvals behind it, and whoever contests the accumulated record carries the burden of proof. Reversal at that point is possible but expensive. The committee removed the earlier Contracts design from the C++20 working draft after adopting it, once design disagreements surfaced that consensus could not resolve [11]. It walked away from P0443R14 [12] after the unified executor design absorbed fourteen revisions of committee direction and was never deployed as designed - a history P4094R1 [13] documents with its costs. Both reversals cost years, and both show the committee can undo even an adopted design once the

case is made. The concern here is narrower and earlier: the cost of reversal accrues before any adoption poll, while the record is built case by case. Reversal stays available - what the accumulation removes is the occasion to decide the architecture before that cost is incurred.

---

4. The Claim Decides Who Owns Runtime-Check Configuration

The bundled architecture claim carries real design stakes: this section shows what the layering decides, then what the deployment record says about the two architectures it chooses between.

Four design consequences follow from the layering claim, each taken from the published texts:

- **It decides who writes guarantees.** Under P3984R0's model, a profile writes the guarantee itself: "A profile cannot change the semantics of a program beyond defining the meaning of some forms of undefined behavior", and its overflow example has the profile choose wraparound, saturation, or an exception [14]. Under P3100R8's Section 4.4, a profile defines nothing. It selects a configuration of the proposal's semantics. The difference is ownership: the profile author writes the guarantee, or the preset author chooses from the proposal's menu.
- **It decides who owns configuration.** P3100R8's Section 7.2 requires that either Labels or the Profiles framework be specified in terms of the other, and the proposal nominates Labels, sketching a profile as "essentially a declaration that expands to [P3400R3] directives". If per-clause review normalizes that arrangement case by case, P3589R2 arrives at its own EWG review already defined as syntax sugar over another proposal's facility.
- **It decides the scope of Profiles.** The alternative future that Section 7.2 offers - Profiles as "an auditing feature rather than a configuration feature" - removes Profiles from configuration entirely. An auditing profile cannot enable anything. It can only reject programs whose configuration, chosen through P3100R8's mechanisms, violates its guarantees. Such a feature is real and possibly useful, but strictly smaller than the framework the Direction Group endorsed in P3970R0 [15].
- **It applies a test to Profiles that the proposal does not apply to itself.** P3100R8's Section 4.3.2 rejects conditional refined behavior: "We also cannot have two different language dialects where the same expression means two different things (overflow or wraparound)." [4] Its Section 5.4 then maps signed integer overflow, for the same expression, to wraparound under ignore, a diagnostic under observe or enforce, an abort under quick-enforce, and undefined behavior under assume - selected per case by an implementation-defined mechanism. Whether five implementation-selected meanings for one expression are themselves dialects is a question per-clause review will never ask, because no single clause raises it. The dialect question implicates both models, P3100's implementation-selected semantics and P3984's profile-defined semantics alike, which is one more reason to settle it in a paper of its own rather than as a byproduct of wording review.

The deployment record speaks to which architecture matches existing practice. For a decade, the named-check-set form the Profiles papers describe has shipped under vendor names:

- **Core Guidelines checkers:** clang-tidy since LLVM 3.8 (March 2016) [16]; MSVC installed by default from VS 2017 [17].
- **Hardened libc++:** since LLVM 18 (March 2024), four named modes, a failed check "reliably terminated", a build setting in Xcode 16 [18] [19]; deployed across Google server-side production at approximately 0.30% average cost [20].
- **libstdc++ assertions:** since GCC 6 (2016); enabled by default for unoptimized builds since GCC 15 [21].

- **MSVC STL hardening:** since VS 2022 17.14 (May 2025); a failed check calls `__fastfail()` "As C++26 Contracts are not yet implemented" [22].

Every one of these is a named, vendor-defined check-set. None routes through the `std::contracts` violation handler adopted for C++26, and none is configured by a Label. libc++ comes closest: as an experimental feature added in LLVM 21 (2025) [23], it lets a translation unit select among four assertion semantics named after the C++26 evaluation semantics (ignore, observe, quick-enforce, enforce), and it lets vendors, though explicitly not users, override the assertion handler [18]. Even there, the semantic is chosen for a build through a vendor macro rather than per assertion through a Label, and the handler is the vendor's rather than the replaceable `std::contracts` one.

By contrast, the proposal's distinctive machinery ships nowhere. The contract-violation runtime adopted for C++26 (P2900R14 [24], adopted February 2025 [25]) has one compiler implementation: GCC 16.1 (April 2026), opt-in, under GCC's blanket experimental C++26 label [26]; Clang reports "No" [27]; MSVC reports "not yet implemented" [22]. Implicit contract assertions have no implementation, and P3100R8 reports no deployment experience of the proposed machinery: the word "experience" does not appear in it [4]. Labels are future tense in the proposal's own text: they "will provide the ability to choose and constrain the evaluation semantic in code" [4]. Two symmetries are worth stating. Neither proposed specification is deployed - the Profiles framework syntax of P3589R2 has no implementation either (the Clang Profiles work of Section 8 implements individual profiles, not that syntax). Both forms also have deep lineage: the named check-set in the vendor deployments above and the compiler-inserted check in the sanitizers and `-ftrapv`/`-fwrapv`, which P3100R8 maps into its model. On both sides, what deployment validates is per-build selection through vendor flags and macros. The in-source per-assertion Labels and the replaceable `std::contracts` handler, the parts that distinguish the proposal, have no such validation. C++26 has, on paper, assigned hardening's future to contract terms - P3471R4 [28] is the first user of the adopted Contracts feature [25] - while every shipping deployment runs the named-guarantee form.

The governing standard comes from the committee's direction paper, P2000R5 [29]: "We change the language and standard library by gradually building on previous work or by providing a better alternative to an existing feature." That standard cuts two ways here, and this analysis does not adjudicate between them: P3100 builds on P2900, the most recently adopted prior work, while the deployed named-guarantee form is the existing practice a Profiles framework would build on. Which reading governs is what Poll 3 asks EWG to weigh. Three committee members applied the existing-practice reading in this exact domain in P3608R0 [30], writing about what to ship in C++26: "the standard library hardening is existing practice, and comes with very positive field experience reports." The existing practice it names is the named-guarantee form, so the reading transfers to the ownership question. Who wrote this standard matters, so here it is: Stroustrup, an author on one side of the dispute examined here, co-authors P2000R5 and P3970R0, and P3608R0 is co-authored by an author of this paper (Section 8). The deployment facts above stand on vendor documentation and do not depend on those papers. Profiles' own direction lineage - P2687R0 [31] (2022) back to the C++ Core Guidelines, announced September 2015 [32] - is weighed exactly as Table 1 weighs the proposal's polls: direction, not adoption.

The layering is a runtime arrangement as well as a diagram. Under P3100R8 a detected core-language violation is routed to the single program-wide contract-violation handler regardless of whether the build selects observe, enforce, or quick-enforce. The log-and-continue shape of the observe semantic already ships at the library level in Bloomberg's `bsls_review` [33]. Whoever owns the handler owns the response, and under P3100R8 that owner is neither the operation nor

any profile - it is the handler. A profile that must guarantee terminate-or-reject for the categories it covers is then defined over a substrate whose configuration can route the response through a handler the profile does not control. Hence the ownership question needs a ballot of its own, separate from the case-by-case review.

One response holds that the two features are complementary and the layering a detail. But complementarity is not symmetric here, and P3100R8 says so. Its Section 7.2 states that one feature must be specified in terms of the other, and the paper places Labels underneath. P3081R1 took the complementary path - it adopted the proposal's API into its own wording - and P3100R8's Section 5.6 then withdrew that API, as Section 3 showed. Complementarity without settled configuration ownership has already produced one stranded dependency. The layering is the decision, not a detail. With this much contested evidence behind it, the question deserves a dedicated ballot, away from 77 wording cases whose technical merits are independent of it.

---

5. No Published Paper Contests the Characterization

The claim of this section is deliberately narrow. Through the 2026-05 pre-Brno mailing (the most recent published mailing at the 2026-07-07 fetch), the search method below found no published WG21 paper that directly contests P3100R8's characterization of Profiles. This is a statement about the public paper record only. It says nothing about whether anyone has objected in committee discussion, which is outside this scope.

Since June 2025, the characterization has been in print - first on slide 53 of P3754R0 [34] (headed "Configurable Profiles", reading "Named configuration presets for the features below"), endorsed by the Sofia diagram poll, then in P3100R4 (August 2025) and every revision since.

Why the absence is structural rather than accidental: a paper with no normative effect never forces anyone to respond. No wording changes what any implementation must do, no forwarding poll on the layering is scheduled, and no individual clause in the review asks the Profiles question. Anyone who objects is objecting to something that, today, does nothing, so the claim moves forward because there is never anything to vote against.

The method, so the absence claim can be re-run: we enumerated all papers in the public open-std 2025 and 2026 mailing directories whose titles match profile, safety, harden, undefined behaviour, UB, erroneous, contract, preset, P3100, or P3754, together with every revision of P3100 and P3754 and the P2687/P3274/P3081/P3589/P3970/P3984 lineage - 121 documents, fetched 2026-07-07. We searched the full text of each for the characterization and its layering language (preset, built on top of, higher-level feature, substrate). We also checked the public GitHub paper tracker and the full WG21 paper index, author by author, for 2025-2026 papers by Stroustrup, Dos Reis, Sutter, and Vandevoorde. A contesting paper with a title outside the keyword net would escape the search. The author scan and the tracker check mitigate that risk. They do not close it.

The result: the characterization occurs only in the proposal's own papers and its companions. P3543R0 [35] (December 2024, co-authored by Doumler and Berne) states that P3081's runtime checks "are already preconditions introduced as implicit preconditions into the language itself by [P3100]". P3599R0 [36] (February 2025) proposes to "restrict [P3081R1] (Profiles) to static checks".

On the Profiles side, the published papers assert an incompatible architecture and do not engage the characterization. P3589R2 contains no occurrence of "contract", "label", or "preset" in any revision - though its latest revision (May 2025) predates the characterization's first publication, so its silence reflects timing, not agreement [37]. P3984R0 post-dates the characterization by eight months and asserts a profile that owns its semantics directly: "For example, signed arithmetic overflow is UB so a profile can define it to be wraparound like unsigned arithmetic (though I wouldn't do that), to be saturated arithmetic, or to throw an exception." The strings "P3100", "contract", and "preset" do not occur anywhere in it [14]. P3970R0, the Direction Group's January 2026 statement, comes nearest: it reports "a stream of uncoordinated proposals to address problems from different perspectives, often not even mentioning Profiles" and reasserts the P3589R2 framework as the way forward - without naming P3100 [15]. Eighteen months ago, in P3608R0, the interaction question itself was identified as open: "We have a very unclear picture on how contracts and profiles should interact and interoperate" [30]. That question came first. The characterization is the proposal's answer, and no published response has appeared.

In parallel, the Profiles papers keep asserting the opposite architecture, as if both could be true. Severance fixes this: the wording review continues on its own track, and the architecture question gets its own poll.

---

6. Objections

Each heading below is an objection this paper expects, stated in its strongest form. Each response draws only on evidence already presented.

"These polls are tautological; wording review obviously covers only its own wording." Then affirming it costs nothing

If the separation were self-evident, Poll 1 would cost nothing and only record what everyone already believes. It is not, because Section 4 shows accumulated wording approvals settling the Section 4.4 layering without a ballot. Interrupting that default now is cheap where reversing it later has cost the committee years.

"Nothing normative changes, so nothing is decided." The architecture still changes

This is the proposal's own strongest defense, and Section 4 concedes its premise: the wording obligates no implementation to do anything. What it decides is architecture, not behavior. P3100R8's Section 7.2 says that if both features are kept, one must be specified in terms of the other, and the proposal makes that choice in Labels' favor. Its Section 5.6 already executed a piece of it, withdrawing an API that another paper's published wording depends on. And the proposed wording amends [defns.undefined] and adds [basic.contract.implicit], so that every case of undefined behavior definitionally carries an implicit precondition assertion that it does not occur (see Section 4, Table 2) [4]. Definitions are the part of the standard every later paper must write against. A change to what undefined behavior *is* does not need runtime effects to have consequences.

"The layering is exposition, not wording, so there is nothing to sever." The exposition already acted on another paper's wording

If the layering claim is purely expository, Poll 1 is free - it asks EWG to affirm what would already be true. But the claim is not purely expository. P3100R8's Section 5.6 withdrew the `detection_mode` enumerators that P3081R2's wording still depends on, leaving that dependency without the mechanism that would satisfy it. Its Section 7.2 states a configuration-ownership requirement and nominates Labels. Exposition that removes the producer of another paper's proposed values and states an ownership requirement is doing design work, and design work belongs in a paper that faces its own poll.

"P3100R8 already is the dedicated paper, so Poll 2 is satisfied." Then the question can be put to a vote

A paper whose title, abstract, and stated subject is the systematic treatment of undefined behavior wording is not a paper dedicated to the question of which feature owns configuration. But suppose it is. Then the layering question is formally before EWG in this review - in which case the chairs can run a poll that states it, so the record shows what was decided. Either the layering is out of scope of the wording review, or it is in scope and votable. Both answers serve the severance - the ask is only that the committee pick one.

A variant holds that the dedicated venue already exists in the separate EWG reviews of P3400R3 and P3589R2: adopt either and configuration ownership is settled without severance. But those reviews poll each mechanism on its own merits. Neither ballot asks which feature owns the other. By the time either adoption poll runs, the characterization will carry the consensus record and case approvals of Section 4 behind it, which is the accumulation Poll 1 and Poll 2 exist to interrupt.

"Severing is Profiles advocacy by other means." The scope statement binds both camps

Severance favors neither architecture: it prevents both from winning without a direct ballot. Poll 1 binds both directions: a Profiles framework paper that accumulated wording approvals would face the same scope statement. Poll 2 requires a dedicated paper from whichever side proposes the layering. The wording review proceeds unblocked. The only thing severed is the claim that has never appeared on a ballot.

"The Profiles relationship is already going to be discussed." A discussion is not a decision

A scheduled discussion, or a reconsideration folded into a broader poll, does not decide which feature owns the configuration of runtime checking. Only a poll on that question does. If such a decision is already intended, the three polls in Section 8 cost nothing: Poll 2 records the intention where it exists and supplies it where it does not. A discussion that reaches no recorded decision leaves the default identified here, settlement through accumulated wording approvals (Section 4), in place.

---

7. Making the Design Commitment Explicit

The concern this paper raises requires no intent on anyone's part. Through the formulations of P3100R8's Sections 4.4 and 7.2, EWG may be unintentionally committing itself to closing the design space for Profiles, one approval at a time. No one needs to choose that outcome for it to arrive - the review only needs to keep running while the claim advances unpolled. The remedy for an unintentional commitment is to make the question intentional: write it out, put it in front of EWG, and let the room decide with open eyes.

Any one of three things suffices. At the review sessions, a minuted statement that approval of wording cases neither adopts nor endorses the layering. A sentence in a future revision of P3100 placing the Profiles relationship out of scope of the wording review. Or the polls below.

Severance delays nothing: as of this writing, the architecture decision is not scheduled at all. Case-by-case review has no final poll on the layering, so the decision arrives only as a side effect after the last case is approved. A dedicated paper and poll can be scheduled directly, and review proceeds in the meantime. Severance creates the venue.

A delegate who supports the wording review and has formed no view on the layering can vote For all three polls. None of them takes a position on the design question. They require only that the question, when it is decided, be decided explicitly.

> ***Poll 1.*** *EWG agrees that approval of individual undefined-behavior cases during P3100's case-by-case wording review does not adopt or endorse the layering described in P3100R8 Section 4.4 and Figure 4, in which Profiles are defined as a higher-level feature building on top of the proposal's tools, among them implicit contract assertions.*

Poll 1 is a scope statement: approving wording cases does not decide the Profiles relationship. It constrains what the approvals mean, not what the review may discuss. The case-by-case sessions proceed exactly as planned. It binds both camps - a Profiles framework paper accumulating wording approvals would face the same ruling. If EWG's view is instead that wording approvals do decide the architecture, better to have the minutes say so than to leave it implied.

> ***Poll 2.*** *Whether the Profiles framework (P3589) or the implicit-contract-assertion machinery of P3100 and its Labels (P3400) governs the configuration of runtime checking of core-language undefined behavior is to be decided by explicit EWG poll on a paper dedicated to that design question, not as a consequence of case-by-case wording approvals.*

Poll 2 is a process commitment: the layering gets its own paper and its own poll, from whichever side proposes it. Its premise is Table 1 - no poll has decided the layering - and its commitment cuts in both directions. Poll 2 also supplies the agreement P3100R8's own Section 7.2 asks for when it says the two features, if both are kept, must be specified one in terms of the other. A paper that requests that agreement cannot object to a poll that records it. If neither the Profiles authors nor the P3100 authors bring the dedicated paper, the authors of this paper will. The commitment is to the ballot itself, whoever writes the paper.

> *Poll 3. When EWG polls on the relationship between the Profiles framework (P3589) and implicit contract assertions (P3100), EWG intends to weigh implementation and deployment experience with both architectures, and expects the dedicated paper of Poll 2 to report that experience.*

Poll 3 is an intent statement about the evidence standard. It gates no review and no ballot. It sets the evidence the dedicated paper is expected to bring, so the standard is fixed before the ballot rather than argued at it. One asymmetry belongs on the table: the named-check-set form has more deployment history today than the implicit-contract-assertion form (Section 4).

If the polls pass, C++ gains a direct venue for the architecture decision and loses nothing: wording review proceeds, implementations proceed, experience accrues, and the layering question arrives at its own ballot with evidence the committee can weigh. If the polls fail, the architecture question continues to be settled by default, one wording case at a time, with no ballot and no reversal mechanism except the kind that cost the committee years in the Contracts and P0443 precedents (Section 3). Whoever writes the dedicated layering proposal builds on this work next.

These three polls are themselves small consensus steps - but each names the question it decides, and a named question can be debated, amended, or voted down. A default cannot.

---

8. Disclosure

The authors provide information and serve at the pleasure of the committee.

Vinnie Falco is the founder of the C++ Alliance, which sponsors a Clang implementation of Profiles. Ville Voutilainen is a longtime WG21 member and a co-author of P3608R0, which Sections 4 and 5 quote, and of other published critiques of the C++26 Contracts process.

This paper takes no position on which architecture is correct. It reports the public deployment record and asks that the ownership question be polled. One of a set in the July 2026 mailing on the runtime checking of core-language undefined behavior, it works only from the published record, and committee-internal documents may contain answers that the record does not. It uses machine-assisted drafting.

References

[1] P4306R0 - "Configuring Runtime Checking: Profiles and Implicit Contract Assertions" (Vinnie Falco, Ville Voutilainen, 2026).

[2] P4310R0 - "Hasta la Vista, Undefined Behavior: Why Implicit Contract Violations Should Terminate" (Vinnie Falco, 2026).

[3] P3400R3 - "Controlling Contract-Assertion Properties" (Joshua Berne, 2026).

[4] P3100R8 - "A framework for systematically addressing undefined behaviour in the C++ Standard" (Timur Doumler, Joshua Berne, 2026).

[5] P3100R2 - "Implicit contract assertions" (Timur Doumler, Joshua Berne, 2025).

[6] P3100R4 - "A framework for systematically addressing undefined behaviour in the C++ Standard" (Timur Doumler, Joshua Berne, 2025).

[7] P3081R1 - "Core safety profiles for C++26" (Herb Sutter, 2025).

[8] P3081R2 - "Core safety profiles for C++26" (Herb Sutter, 2025).

[9] P3656R1 - "Initial draft proposal for core language UB white paper: Process and major work items" (Herb Sutter, Gašper Ažman, 2025).

[10] cplusplus/papers issue #1901 - public WG21 paper-tracker issue for P3100; records the EWG Brno straw poll of 2026-06-10 (question text and tally 16/15/6/2/0, result consensus).

[11] Trip report: Summer ISO C++ standards meeting (Cologne) - "Contracts moved from draft C++20 to a new Study Group" (Herb Sutter, 2019).

[12] P0443R14 - "A Unified Executors Proposal for C++" (Jared Hoberock, Michael Garland, Chris Kohlhoff, Chris Mysen, Carter Edwards, Gordon Brown, Daisy Hollman, et al., 2020).

[13] P4094R1 - "Info: The Unification of Executors and P0443" (Vinnie Falco, 2026).

[14] P3984R0 - "A type-safety profile" (Bjarne Stroustrup, 2026).

[15] P3970R0 - "Profiles and Safety: a call to action" (David Vandevoorde, Jeff Garland, Paul E. McKenney, Roger Orr, Bjarne Stroustrup, Michael Wong, 2026).

[16] clang-tidy checks, LLVM 3.8 - "cppcoreguidelines-pro-bounds-array-to-pointer-decay" (LLVM Project, 2016).

[17] Using the C++ Core Guidelines checkers - "Use the C++ Core Guidelines checkers" (Microsoft Learn, retrieved 2026).

[18] libc++ Hardening Modes - "Hardening Modes" (LLVM Project, retrieved 2026).

[19] C++ Language Support - "C++ Language Support" (Apple Developer, retrieved 2026).

[20] Retrofitting spatial safety to hundreds of millions of lines of C++ - "Retrofitting spatial safety to hundreds of millions of lines of C++" (Alex Rebert, Max Shavrick, Kinuko Yasuda, Google Security Blog, 2024).

[21] libstdc++ macros documentation - "Macros" (GCC, retrieved 2026); API history at libstdc++ API evolution.

[22] microsoft/STL VS 2022 changelog - "VS 2022 Changelog" (Microsoft STL team, retrieved 2026); hardening details at STL Hardening wiki.

[23] Libc++ 21 Release Notes - "Libc++ 21.1.0 Release Notes", assertion-semantics section (LLVM Project, 2025).

[24] P2900R14 - "Contracts for C++" (Joshua Berne, Timur Doumler, Andrzej Krzemieński, 2025).

[25] Trip report: February 2025 ISO C++ standards meeting (Hagenberg, Austria) - (Herb Sutter, 2025).

[26] GCC 16 release notes - "GCC 16 Release Series: Changes, New Features, and Fixes" (GCC, 2026); flag documentation in the GCC 16.1 manual; experimental C++26 label at C++ Standards Support in GCC.

[27] C++ Support in Clang - "C++ Support in Clang" (LLVM Project, retrieved 2026).

[28] P3471R4 - "Standard library hardening" (Konstantin Varlamov, Louis Dionne, 2025).

[29] P2000R5 - "Direction for ISO C++" (Jeff Garland, Paul E. McKenney, Roger Orr, Bjarne Stroustrup, David Vandevoorde, Michael Wong, 2026).

[30] P3608R0 - "Contracts and profiles: what can we reasonably ship in C++26" (Ville Voutilainen, Jonathan Wakely, Gabriel Dos Reis, 2025).

[31] P2687R0 - "Design Alternatives for Type-and-Resource Safe C++" (Bjarne Stroustrup, Gabriel Dos Reis, 2022).

[32] Bjarne Stroustrup announces C++ Core Guidelines - "Bjarne Stroustrup announces C++ Core Guidelines" (isocpp.org, 2015).

[33] bsls_review - "bsls_review component documentation": the default review handler logs and returns rather than aborting, so checks can be added to production software without changing its behavior (Bloomberg BDE, retrieved 2026).

[34] P3754R0 - "Slides for P3100R2 presentation to EWG" (Timur Doumler, 2025).

[35] P3543R0 - "Response to Core Safety Profiles (P3081)" (Mungo Gill, Corentin Jabot, John Lakos, Joshua Berne, Timur Doumler, 2024).

[36] P3599R0 - "Initial Implicit Contract Assertions" (Joshua Berne, Timur Doumler, 2025).

[37] P3589R2 - "C++ Profiles: The Framework" (Gabriel Dos Reis, 2025).