Addressed but Unresolved: P3846R1's Eighteen Responses on C++26 Contract Assertions

Document Number:	P4272R0
Date:	2026-09-03
Intent:	Inform
Audience:	EWG

Reply-to: Vinnie Falco vinnie.falco@gmail.com

Table of Contents

Abstract

Sixteen of the eighteen objections to C++26 contract assertions were never settled. None was closed by a draft change the objector accepted, by an explanation whose evidence supported its conclusion, or by a recorded decision with stated reasons, and no attempt to settle them is visible on the public record either.

This paper does not ask whether the eighteen objections are correct. It asks whether they were settled, and settled has a definition: one of the three forms the ISO/IEC Directives and disposition practice recognize, a draft change with the commenter's confirmation, an explanation whose evidence suffices for its conclusion, or a recorded decision with a stated rationale. The assessment tests the closure claim in P3846R1 against the public record at two cutoff dates. It rates each response for its support on the record, and separately records whether the objection was settled in one of those three forms. *Addressed* is P3846R1's word; *resolution* is the Directives', and what the Directives oblige is a good-faith attempt at it; *settled* is this paper's word for an objection resolved in one of the three forms.

P3846R1, "C++26 Contract Assertions, Reasserted," answers eighteen concerns about the contract assertions facility adopted into the C++26 working draft, and its abstract asserts that the objections were addressed in subsequent responses and extensively discussed in EWG. On support, the record is in the authors' favor: Two responses are Supported, six Substantially Supported, ten Mixed, and none Unsupported or Contradicted, though fifteen responses contain a material assertion the record does not support, four of them contradicted by it. On settlement, sixteen of the eighteen objections were not settled: two Settled, fifteen Replied without settlement, one Not reached. The support result is why this is not a finding of bad faith or poor work; the settlement result is the finding, and it holds because a reply is not a settlement. The reflector discussion shows the distinction in operation: Live objections are directed to P3846, and prior votes, roadmap boundaries, timing, or future work are invoked as reasons to retain the C++26 design unchanged.

The three forms of resolution are not theoretical. All three occurred in this dispute, twice by a response and once by a draft change whose authors record the commenters' confirmation. The distribution is therefore a finding about the responses rather than the rubric: Substantive argument was produced, and closure was not. The assessment then states what a discharged reconciliation duty leaves on the record, drawn from practice in force in WG21's Library Evolution group, WG14, and ASTM, finds that record absent from the adoption arc in both the public sources and the member-accessible session notes, and sets the arc beside WG21's own P2300 record, where a written finding was followed by acceptance and here the absence of one was followed by escalation at every stage. Two findings result, and neither entails the other: Sixteen objections are unsettled, which a future response paper can change one row at a time by supplying the evidence Table 3 names; and no attempt at reconciliation is visible on the record, which no response paper can change, because that record is the committee's to make.

Revision History

R0: September 2026

- Initial version.

---

1. Introduction

P3846R1, [1] "C++26 Contract Assertions, Reasserted," answers eighteen concerns raised about P2900R14, [2] the contract assertions facility adopted into the C++26 working draft at the February 2025 Hagenberg meeting. Its abstract states, "Almost all objections are repetitions of those raised in earlier papers, addressed in subsequent responses, and extensively discussed in EWG" (p. 1), and qualifies the claim: "Some of the concerns are legitimate yet unavoidable with any viable assertion facility. Others reflect misunderstandings of the proposal, leading to inaccurate observations" (p. 1). The closure claim is tested twice: Each of the eighteen responses is assessed against the public record, then measured against the resolution the ISO/IEC Directives oblige the process to attempt. *Addressed* is P3846R1's word, and *resolution* is the Directives'; Section 2 sets out why the two are held to the same measure. The abstract's qualification is answered as an objection in Section 8.

P2900 provides contract assertions in declaration position (preconditions, postconditions, and `contract_assert`) with four evaluation semantics selectable per translation unit and a global contract-violation handler invoked when a checked assertion fails. P3846R1 is the consolidated response by twenty-two authors, including two of the three authors of P2900R14. The concerns it answers come from the objecting papers P3835R0, [3] P3829R0, [4] P3849R0, [5] P3506R0, [6] and P3878R0, [7] and from national body comments on the C++26 draft whose dispositions the Kona November 2025 minutes record. [8]

This paper makes seven contributions.

1. The resolution obligation of the ISO/IEC Directives, stated with its scope, applied to P3846R1's closure claim (Section 2).
2. A three-field assessment method at two cutoff dates, separating support for a response from defects in its subclaims from resolution of the objection (Section 3).
3. The eighteen-concern assessment (Sections 4 and 5): two responses Supported, six Substantially Supported, ten Mixed, none Unsupported or Contradicted; fifteen responses with a material Unsupported Subclaim; two objections Settled, fifteen Replied without settlement, one Not reached, and, for each of the sixteen, a statement of what would have settled it (Table 3).
4. A concern-level and thematic record of the P3846R1 authors' own words, showing prior responses, polls, roadmap boundaries, timing, and future work functioning as closure (Sections 5 and 7).
5. Four clarifications to the public record: a. The 2016 lost-optimization report now numbered LLVM issue 28170 was filed as Bugzilla 27796 and renumbered when the LLVM tracker migrated to GitHub; citations to either number refer to the same report. [9] b. The Kona minutes N5031 are a 2025 document; their own cover misprints the date as 2024-11-28. [8] c. P3626R0 is the alternative wording prepared by a coauthor of P2900R14 and P3846R1; P2899R1 records that it "provided the wording diff" as supplementary material for EWG's Hagenberg discussion, and it is not an independent rival proposal. [10] [20] d. P3097R0 was merged into P2900R8, [11] a revision of the proposal, never into the C++26 working draft. [12]

6. A description of the record a discharged reconciliation duty produces, each element located in practice in force today, with a WG21 instance of the record being made, and the Contracts arc set beside that instance stage by stage (Section 9, Table 5).
7. The assignment of the reconciliation duty to the convenership, and the absence, across the public sources and the member-accessible session notes searched, of any reconciliation record during the adoption arc, read against that description; a question of office rather than of authorship, with the search scope and the case for a strong executive set out in Section 10.

Three assumptions govern the assessment. First, a paper's printed date fixes what its authors could have known, so a source dated after 2025-11-03 neither falsifies nor supports a sentence written by that day.

This first assumption is narrower than it may appear. A later source can still answer an objection: If a question is asked on Monday and a paper published on Tuesday answers it, the question is answered, and the objection's state is recorded accordingly at the declared cutoffs. What the Tuesday paper cannot do is make the Monday sentence have been supported when it was written. The distinction is between auditing an artifact and adjudicating a question. The audit asks whether each sentence in P3846R1 was supported by the record available to its authors; whether an objection now stands or falls is a separate question, and Section 3's evidence rule says so directly, that later evidence "can support a future revision."

Second, a committee vote establishes procedural disposition without settling a technical question. Third, the admissible evidence is material that is on the record and retrievable, whether world-readable or accessible to committee members; a claim resting on discussion that leaves no retrievable record is recorded as unsourced rather than as false.

A fourth assumption governs the other three and the paper's title. This paper does not ask whether the eighteen objections are correct, and nothing below should be read as a verdict on their merits. It asks whether they were settled, in one of the three forms Section 2 sets out, and it finds that sixteen were not. An objection can be wrong and settled, by a recorded decision with reasons; it can be right and unsettled, by a reply that leaves its mechanism where it found it. The two questions are independent, and only the second is asked here.

---

2. What the Process Says Consensus Means

Nothing in this paper depends on the ISO/IEC Directives binding a working-group paper. The Directives are cited as the process's own account of what consensus is and what reaching it requires, and the record is measured against that account. A reader who holds that no duty attached below the ballot stage loses none of Sections 4 through 9, which describe what the record contains and what followed from it.

The ISO/IEC Directives define consensus around resolution. The foreword principles state [13]:

> *Consensus, which requires the resolution of substantial objections, is an essential procedural principle and a necessary condition for the preparation of International Standards that will be accepted and widely used. Although it is necessary for the technical work to progress speedily, sufficient time is required before the approval stage for the discussion, negotiation and resolution of significant technical disagreements.*

Clause 2.5.6 adopts the ISO/IEC Guide 2 definition, which characterizes consensus as a process "seeking to take into account the views of all parties concerned and to reconcile any conflicting arguments," and adds: "If the leadership determines that there is a sustained opposition, it is required to try and resolve it in good faith." [13] Clause 2.6.5 places the two duties side by side: "Committees are required to respond to all comments received," and "Every attempt shall be made to resolve negative votes." [13] Response is the floor. The obligation above it is a good-faith attempt at resolution: seeking to reconcile, trying to resolve, and making every attempt. The obligation is to attempt rather than to succeed, and an attempt at reconciliation leaves a record.

Disposition practice contains no middle state. WG21's own disposition of comments for the C++20 committee draft, N4858, records three technical outcomes,

1. "Accepted"
2. "Accepted with Modification," naming the adopted paper that resolves the comment, and
3. "Rejected. There was no consensus to adopt this change,"

plus an editorial variant, "Accepted - Editorial," for comments routed to the editor. [14] ISO/TC 211's good-practice guidance for the enquiry stage instructs editors to elaborate on "Accepted in principle" and "Not accepted" dispositions "to make sure you do not get the same comment the next ballot or even a No vote in the FDIS ballot." [15] No recognized category records that a comment was replied to but not settled. For the current cycle, the SC 22 summary of voting for the C++26 committee draft instructs "review and resolution of the comments" and directs the Project Editor to prepare "an approved Disposition of Comments document and a revised text for further processing." [16]

Peer processes draw the same response-resolution distinction. The IETF's consensus doctrine holds that "the existence of the unaddressed open issue, not the number of people" is determinative and that a consensus finding owes the objector "a reasoned explanation to the person(s) raising the issue of why their concern is not going to be accommodated." [17] ANSI's Essential Requirements define a resolved comment as one where "the negative commenter accepts the proposed resolution of his/her comment." [18] W3C's process defines "formally addressed" as a public, substantive response whose adequacy "is measured against what a W3C reviewer would generally consider to be technically sound." [19] Three peer processes impose an adequacy test on the response itself; none recognizes a reply that leaves the objection standing as a disposition. Each test is also stated from the objector's side, in terms of what the objector can read, accept, or contest.

These provisions bind the standards process at two levels, the national body ballot and the plenary. P3846R1 is a working group paper, and no directive obliges it to meet the ballot-stage obligation as a document. The definition of resolution nonetheless supplies the test for its claim, for two reasons. First, its abstract asserts that the objections were addressed, and "addressed" must mean something; the Directives define what the standards process means by resolving an objection. Second, the objections it answers include national body comments on the C++26 draft, the same objections now before the national bodies at the Draft International Standard (DIS) ballot, where the resolution obligation governs outright.

Under that obligation, a response to a formal objection resolves it in one of three ways.

1. **Resolution by change.** The draft text is modified, and the commenter confirms the change resolves the concern.

2. **Resolution by explanation.** The committee gives a substantive, reasoned response whose evidence suffices for its stated conclusion and engages the objection's central mechanism.
3. **Disposition by recorded decision.** The committee rejects the concern by a recorded consensus with a stated rationale, converting the objection into a voted outcome.

A response that explains a tradeoff while leaving a material technical issue open does none of the three. It is a reply, and the Directives' vocabulary already has a place for replies: the floor. A companion paper, P4195R0, states the same test from the record's side, as three properties a later reader can check without having been in the room: the objection stated in terms its holder would recognize, a response addressing its substance, and a finding by the leadership recorded at the time. [109] The third of those is this paper's Form 3, and an objection answered and then overruled in a recorded finding satisfies both formulations. The assessment below tests each response against these three forms; whether the authors' rationale paper, P2899R1, [20] itself satisfies the third form is taken up as an expected objection in Section 8.

---

3. Method: Three Independent Fields at Two Cutoffs

Two cutoff dates fix the public record against which each claim is tested, and three fields score every concern; the fields answer different questions and are never combined. Public statements after both cutoffs enter only as later corroboration of how the response corpus continued to function, are identified as such where used, and change no score.

The audited artifact is the P3846R1 PDF published in the April 2026 WG21 mailing [1]: 494,439 bytes, SHA-256 `0cbbdc9c27987d5694b5d4f6d48d97c3244d8c40a547225ce79cf060bd46035c`. The artifact prints "Date: 2025-11-03" on its title page; its PDF metadata reads 2026-03-23, its tracking issue records "Authors provided updated version" that day, [21] and the official 2026 index dates it 2026-03-23. [22] The printed date is the substantive cutoff: Sources dated on or before 2025-11-03 test whether the responses were accurate when written. The build date is the publication-state cutoff: Sources dated between the two test whether the artifact was current when supplied. Evidence after 2026-03-23 enters no score. No public artifact establishes that text byte-identical to the audited PDF circulated on the printed date: The 2025 index lists no P3846R1, [23] and the Internet Archive holds no capture of any P3846R1 before June 2026. The PDF thus bears a title-page date four and a half months earlier than the date it was built and supplied. No allegation is made that the earlier date was chosen to mislead; a drafting date left unchanged when a document is revised is common and innocent. The discrepancy matters only because it makes "when was this sentence written" ambiguous, and the ratings are therefore taken against both dates rather than one. Neither date is an expiration date for answering an objection; both are cutoffs for judging whether a printed sentence was supported at the time.

The quotation record serves a narrower purpose than the scores. Statements by P3846R1 authors are selected when they show an objection acknowledged alongside a prior-response pointer, poll, consensus claim, roadmap boundary, timing rule, or future-work answer. They establish how the response corpus was used; they do not establish the authors' motives. A statement after 2026-03-23 may corroborate that later function or describe later work; it cannot supply missing support for P3846R1 or retroactively resolve an objection at either cutoff.

In Overall Support, the complete response is rated, on five values, against the record at the cutoffs.

1. Supported
2. Substantially Supported
3. Mixed, where Mixed means material support and material weakness both affect the central response
4. Unsupported
5. Contradicted

The Unsupported Subclaim flag is a separate yes/no field, set when the response contains a specific material assertion (factual, causal, quantitative, or historical) that lacks support or is false, regardless of the complete response's rating. The Settled field answers a third question: whether the objection was settled in one of Section 2's three forms, which for a single response means whether the response meets the original objection on its own terms. It takes three values, and only the first is a settlement. "Settled" means the supported response meets the stated objection, or a recorded decision with reasons or a confirmed draft change disposes of it. "Replied" means the response explains a tradeoff or documents a procedure while leaving a material technical issue open; the objection is not settled, and a reply is on the record. "Not reached" means the response does not meet the objection's central mechanism or evidence; the objection is not settled, and no reply reaches it. A committee vote establishes procedural disposition; it does not by itself move an objection from Not reached or Replied to Settled unless the decision is recorded with its reasons. The three fields are kept separate because combining them produces two characteristic errors, treating one incorrect sentence as the failure of an entire response and treating a committee decision as proof of a technical proposition.

The evidence rule limits both sides alike. Expert testimony may support a qualitative judgment when public and attributed but cannot support an unattributed quantitative comparison. A coding rule proves that a risk is recognized, without proving how often it materializes. Evidence made public after the build-date cutoff cannot retroactively support a printed sentence; it can support a future revision, which is the audit distinction Section 1 draws. Because no published minutes record subgroup straw-poll tallies, poll tallies come from the chair-posted threads on the public cplusplus/papers tracker. The Hagenberg minutes refer readers to "polls on the P2900 tracker" and to the "github tracker" without naming the repository, [24] and the rationale paper corroborates each tally where that paper records the same poll. [20] The EWG session notes for Wrocław and Hagenberg on the committee wiki, accessible to members, were also examined. [107] [108] They record the discussion speaker by speaker and, for Hagenberg, the closing tallies; they contain no finding by the chair in words beside any tally, and Section 10 states what they do contain. Reflector material from the isocpp.org list archive is paraphrased rather than quoted, and no participant is named; posts are cited by number so that the paraphrase can be checked against the archive. The evidence rule is the author's own construction, and the author has a material stake in the question under assessment, as the Disclosure section states. If the author's rule is rejected, the quoted sentences and cited artifacts stand independently of the rating scheme, and the Settled column rests on the Directives' definition rather than on the author's.

Three limitations bound what the eighteen-concern distributions can mean; Section 5 states concern-level caveats in place.

1. No public artifact establishes the November text's byte identity with the March PDF.
2. The search for one claimed compiler prototype covered public sources only. The gcc.gnu.org archives sit behind an interactive bot filter and were read through a public mirror that is complete for the gcc-patches and gcc-bugs lists; private WG21 venues, in which a prototype may have been shared, are outside the public record by definition.

3. The ratings are applications of a stated rule to cited evidence rather than measurements. Each verdict names the sentence and the source it turns on, so a reviewer who disagrees can say which. Section 8 works through what happens if every contested boundary moves, by one category and by two.

---

4. The Scorecard: Responses Largely Supported, Objections Largely Unsettled

Table 1. Assessment of the eighteen P3846R1 responses. Overall Support rates the complete response against the public record at the cutoffs of Section 3. Unsupported Subclaim records whether the response contains a material unsupported factual, causal, quantitative, or historical assertion. Settled records whether the objection was settled in one of Section 2's three forms; "No, replied" and "No, not reached" are both unsettled. The three fields are independent by construction.

Concern	Overall Support	Unsupported Subclaim	Settled
1. Safety and non-ignorable checks	Mixed	Yes	No, replied
2. Cross-translation-unit semantics	Mixed	Yes	No, replied
3. Dependency management	Mixed	Yes	No, replied
4. One Definition Rule	Substantially Supported	Yes	No, replied
5. Modules	Supported	No	No, replied
6. Implementation-defined behavior	Mixed	Yes	No, replied
7. Uncheckable guidance	Substantially Supported	Yes	No, replied
8. Constification	Substantially Supported	No	Yes
9. Global violation handler	Substantially Supported	Yes	No, replied
10. Consecutive assertions	Supported	No	Yes
11. Predicate exceptions	Mixed	Yes	No, not reached
12. Static analysis	Mixed	Yes	No, replied
13. Complexity	Mixed	Yes	No, replied
14. Missing features	Substantially Supported	Yes	No, replied
15. Future features	Mixed	Yes	No, replied
16. Decomposition	Mixed	Yes	No, replied
17. Deployment experience	Substantially Supported	Yes	No, replied
18. Library hardening	Mixed	Yes	No, replied

The support counts are two Supported, six Substantially Supported, ten Mixed, zero Unsupported, and zero Contradicted; fifteen responses receive the Unsupported Subclaim flag. The settlement counts are two Settled and sixteen not settled, of which fifteen are Replied and one is Not reached. The two axes diverge. On support the scores favor the authors of P3846R1, with eight responses rated favorably against ten Mixed. On settlement, sixteen objections stand where the responses found them. The divergence is the finding: The responses are, in the main, accurate replies, and a reply is not a settlement. Neither column is a verdict on whether the objections are correct, since the support column rates responses rather than objections and the Settled column records the state of each objection against the Directives' definition.

5. The Eighteen Concerns

Concern 5 shows the two axes coming apart in the simplest form, and it is worth reading first for that reason. The response is Supported and contains no unsupported sentence. It also settles nothing, because the objection asked who controls the evaluation semantic across a module boundary and the response, accurately, offered an avenue "in principle" and called it "only a partial solution." A response can be true in every sentence and leave the objection where it found it. That is the pattern the sixteen unsettled objections share, and it is the pattern the title names: addressed, and not settled. The flags in Table 2 are a separate finding, about accuracy, and a response can receive a flag and still be closer to settlement than Concern 5 is.

5.1 Two fully supported responses

Concern 5: Modules. The objection is that P2900 does not work well with modules; P3835R0 asks whether modules could address the configuration of contract-evaluation semantics. [3] The response is bounded: "In principle, inline functions in a BMI could carry additional information, such as contract-evaluation semantics" (p. 16), and the same paragraph limits the claim to "only a partial solution to the broader problem" (p. 16). GCC's module serialization change implements that mechanism, streaming references to the outlined precondition and postcondition helpers and appending a boolean to the built-module-interface dialect string; it was merged into the GCC master branch on 2026-01-28. [25] No evaluation semantic is written to or read from a module interface anywhere in the change, so the practical question of who controls the semantic across module boundaries remains open.

> *The interaction between P2900 and modules was previously raised in [P3573R0], responded to in [P3591R0], and further discussed in EWG in Hagenberg. No new information has been presented since. (P3846R1)* [1]

The Discussion Status cites a previous response, prior discussion, and absence of new information, while the response itself calls modules "only a partial solution."

Verdict. Overall Support: Supported. Unsupported Subclaim: No. Settled: No, replied. The architectural avenue exists at the level claimed, and the practical question remains.

Concern 10: Consecutive assertions. The objection is that observing consecutive assertions is dangerous because an earlier assertion may be a precondition for safely evaluating a later one. The response supplies a mechanism, a counterexample, and a committee record: "The idiomatic solution is to combine dependent predicates into a single assertion, thus avoiding the risk of evaluating the second condition after the first fails" (p. 23), and states the residual risk. Short-circuit conjunction works for the canonical case. The Contracts Study Group (SG21) polled the proposed alternative, automatically skipping subsequent assertions, on 2025-02-06 and declined it: SF 0, F 0, N 1, A 13, SA 7, "Consensus against," [26] corroborated by the rationale paper. [20] The proposal itself states that no general method distinguishes related predicates from unrelated ones. [27]

"The observe semantic is an indispensable tool when introducing new contract assertions into existing code, but continuing past a failed assertion always comes with a risk." (P3846R1) [1] In the reflector discussion, the same issue drew a pointer back to the paper's own Concern 10 rather than a fresh answer (SG15 post 2744). [28]

Here the reference back to P3846 accompanies a working mitigation and a recorded decision. This is the control case, where the process record and a response that meets the objection's mechanism coincide.

Verdict. Overall Support: Supported. Unsupported Subclaim: No. Settled: Yes, a working mitigation for the canonical case, a counterexample against the alternative, and a recorded committee decision declining that alternative.

5.2 Six supported central claims, each with a material limit, one of them resolving its objection

Concern 4: One Definition Rule. The objection is that P2900 violates the spirit of the One Definition Rule. The response classifies the reported failure as a general compiler defect: "both Clang (LLVMPR26774) and GCC (GCCBug70018) disabled them nearly a decade ago" (p. 15), and the behavior reported in P3829R0 is "a regression of the same issue in GCC 14, entirely unrelated to contract assertions" (p. 15). The classification holds: Clang's fix was committed on 2016-04-08, [29] GCC's appeared in the GCC 7 series, [30] and the 2025 bug's reproducer contains no contract assertion and no contracts flags. [31] The performance dismissal does not hold: The response states that such concerns are "equally unfounded" and that "Clang made this tradeoff long ago without user complaints" (p. 15), yet the cited bug contains Jan Hubička's report that "these optimisations may have a large performance impact," with one workload where the optimization "improved jpeg-xl encoding speed by 47%." [31] Thirty-nine days after the Clang fix, Warren Ristow filed the report now numbered LLVM issue 28170, resolved in 2018 without relaxing the conservatism. [9] The same bug's comment 9 enumerates "contract checking mode" among the defect's triggers, and the GCC contracts implementation includes a dedicated default-on workaround, merged with the note that it suffices "while a suitable general fix is evaluated." [31] [32] Contracts exposed the defect and required operational mitigation, which does not make the defect a contracts design issue.

> *EWG, in Hagenberg, considered the general concerns regarding mixed mode and rejected altering the ODR for contract assertions. The specific issue with interprocedural optimisation in GCC was discussed only on the reflector, where it was identified as a compiler bug unrelated to P2900. (P3846R1)* [1]

The Discussion Status pairs an EWG decision on general ODR policy with a reflector-side classification of the specific GCC mechanism. The authors' rationale paper describes the EWG decision in its own terms: "EWG took a deliberately vaguely worded poll to gauge the interest of the room to 'do something about it'; the result was consensus against" (P2900: add ODR to contracts, SF 6, F 5, N 10, A 25, SA 17) (P2899R1, Section 3.5.11). [20] The decision the Discussion Status invokes is a poll its own proposers describe as deliberately vague.

Verdict. Overall Support: Substantially Supported. Unsupported Subclaim: Yes, "equally unfounded," contradicted by a source the response cites. Settled: No, replied.

Concern 7: Uncheckable guidance. The objection is that P2900 relies on guidelines the compiler cannot check. The response separates the design question from the frequency with which the problem occurs. The constraint "is not specific to P2900's design" (p. 19), enforcing it would reject most useful expressions, [33] and "Decades of experience with these facilities have shown that destructive side effects from predicates are easily identified during development and testing and are rarely an issue" (p. 19). The cited analysis supports the design half. [33] The frequency half supplies no data, survey, or defect study, and

the counter-evidence is equally bounded. CERT's PRE31-C rule recognizes the bug class while labeling its likelihood "Unlikely" in its risk table. [34] The two static-analysis checks in this class target other languages: SonarQube S3346 is a C# rule and PVS-Studio V6055 is a Java diagnostic. [35] [36]

> *The desire to produce rules that a compiler can enforce to restrict predicates to only those that are nonproblematic has been put forth in papers such as [P2680R1], [P3285R0], and [P3362R0]. These papers have been given ample committee time and not achieved consensus. No new information has been presented since. (P3846R1)* [1]

The Discussion Status records committee time, failed consensus, and absence of new information; none of the three supplies the frequency evidence on which the response's practical dismissal depends. The rationale paper describes that committee time. Of the Wrocław poll on strict predicates (SF 10, F 6, N 3, A 14, SA 16): "EWG had consensus against pursuing this direction, although sustained opposition to that decision remained," with the recorded result "Consensus against, but P2900 would be in danger of failure in plenary." Of the Hagenberg poll "P2900: reduce the amount of Undefined Behavior in contracts" (SF 9, F 7, N 9, A 22, SA 19): "The group took a poll to do 'something' about this concern" (P2899R1, Section 3.6.1). [20] Both descriptions come from the proposers, and both concern polls on the general direction rather than on this objection's mechanism. Committee time was spent, but the disposition the Discussion Status invokes is a pair of polls their own authors characterize as unfocused.

Verdict. Overall Support: Substantially Supported. Unsupported Subclaim: Yes, "rarely an issue." Settled: No, replied.

Concern 8: Constification. The objection is that const-ification changes the meaning of predicates, complicates teaching, and obstructs automatic assertion insertion. The response reports that "No compelling real-world examples of correct assertions rendered incorrect by const-ification have been produced" (p. 20) and that const-ification "revealed genuine bugs in existing libraries" (p. 20). The migration record supports it: Applied to BDE, the experiment found six assignment-versus-equality defects, [37] and applied to LLVM, approximately seventy-five const-correctness defects before about 98.5 percent of assertions compiled. [38] EWG retained the feature through two removal polls, at Wrocław in November 2024 and at Hagenberg in February 2025, with the wider margin against the stronger question. [39] [40]

The rationale paper records the decision chain with its reasons: SG21 adopted the rule on 2023-12-14 (SF 6, F 10, N 3, A 0, SA 0). EWG at St. Louis polled removal at SF 9, F 5, N 4, A 10, SA 5, "No consensus for change but a strong divide." SG21 declined two removal proposals on 2024-05-16 and voted to retain on 2024-10-10 (SF 3, F 1, N 2, A 10, SA 8, consensus against removal) while extending the rule to all variables (SF 6, F 14, N 2, A 0, SA 3). EWG then voted against removal at Wrocław (SF 10, F 4, N 9, A 19, SA 12) and at Hagenberg (SF 9, F 7, N 6, A 37, SA 14). The same section states both tradeoffs the objection names, non-const-correct APIs and overload resolution differing inside and outside predicates; notes that the confusion concern was weighed as early as P2388R0; and records the migration studies. [20] Part of the objection survives. The BDE result is attributable to const-ification and the restricted predicate grammar together, [37] and the teachability question is answered by analysis rather than by demonstrated harmlessness.

> *In Wrocław, EWG reached consensus against removing const-ification, which was reaffirmed in Hagenberg. The concern in [CZ 4-058] that const-ification could increase the difficulty of automatic assertion insertion by tooling and that const-ification could be replaced with erroneous behaviour are new. Otherwise, no new information has been presented since Wrocław. (P3846R1)* [1]

The Discussion Status acknowledges two new forms of the objection and records the earlier retention decisions, which dispose of the objection as stated. The two new forms, automatic insertion by tooling and replacement with erroneous behavior, are residual: They postdate the decisions and are not yet answered, and they do not reopen the question the decisions settled. Besides Concern 10, this is the one concern where Section 2's second and third resolution forms are both on the record.

Verdict. Overall Support: Substantially Supported. Unsupported Subclaim: No. Settled: Yes, a mechanism with a stated purpose, the objection's own tradeoffs weighed on the record, repeated recorded decisions declining the alternative, and migration evidence.

Concern 9: Global violation handler. The objection is that a global contract-violation handler is problematic. The response grounds its analogy in the standard library: "C++ already includes several global handlers for this purpose (e.g., std::set_new_handler, std::set_terminate, signal handlers), and similar mechanisms are widely and successfully used in major frameworks, such as Qt and in game engines" (pp. 22-23). The standard-library half stands, and the production history is documented in a source the response cites: BDE deployed user-provided violation handlers in 2004 and continued using them. [41] The Qt and game-engine half names no deployments and no outcomes.

> *The global contract-violation handler was adopted into P2900 when SG21 approved [P2811R7], and it had consensus in EWG. Local violation handlers have been proposed in [P3400R1] as a post-C++26 extension. No new information has been presented since. (P3846R1)* [1]

The Discussion Status cites adoption history and a future extension. The deployment claim remains unsupported, and the case for local control remains unaddressed.

Verdict. Overall Support: Substantially Supported. Unsupported Subclaim: Yes, the Qt and game-engine success claim. Settled: No, replied.

Concern 14: Missing features. The objection is that P2900 lacks important features. The response is an incremental-delivery argument with one categorical sentence: "All the requested features have been discussed in various papers; no proposals that included them gained consensus in EWG" (p. 28). One page later, the same section records the exception that falsifies the sentence: "pre and post on virtual functions do have a proposal ([P3097R0]) that is fully specified, has been reviewed and approved with strong consensus in EWG, has been reviewed by CWG, has been implemented in GCC, and could be re-added to the C++ working draft any time EWG wishes to do so" (pp. 29-30). EWG at St. Louis in 2024 polled merging P3097R0 into P2900: SF 18, F 15, N 5, A 1, SA 2 [42]; the tracker comment records no outcome, and the rationale paper records the result as

consensus. [20] The feature entered P2900R8 (a revision of the proposal), never the working draft, and was struck at Hagenberg before P2900R14 entered the draft. [40] The rationale paper records both events: "[S]upport for virtual functions was added to the Contracts proposal in revision [P2900R8]" after the St. Louis poll, and Hagenberg's "disallow pre/post contracts on virtual functions entirely" (SF 20, F 24, N 13, A 14, SA 2) "reversed EWG's previous decision in St. Louis" (Section 3.3.2). [20] The incremental-delivery argument around the inaccurate sentence is substantive and supports extensibility without giving users the capabilities in C++26.

"The lack of syntactic control for contract assertions is certainly a concern, but the ability to introduce such things is a layer of complexity that has been explicitly left to future proposals that build on top of [P2900R13]" (P3591R0). [43] The same use case was later answered on the reflector by pointing to P3400 and adding that SG21 and EWG had spent considerable time polling which features to target in the initial MVP and which to leave until later and that the Contracts feature in the working draft is the result of such decision-making (SG15 post 2980). [44]

The P3591R0 sentence acknowledges the gap and assigns it to future work; the reflector answer gives the feature-selection polls as the reason the C++26 design stands.

Verdict. Overall Support: Substantially Supported. Unsupported Subclaim: Yes, the categorical consensus sentence, which is false. Settled: No, replied.

Concern 17: Deployment experience. The objection is that P2900 has insufficient deployment experience. The response accepts the evidentiary expectation the objection invokes and states its strongest sentence: "Expecting a reasonable level of implementation experience before standardising a novel language feature is good engineering practice that we strongly support. P2900 has been fully implemented in two major compilers" (p. 33). Eight lines later, the same page qualifies both implementations as "nearly complete" with "upstreaming in progress" (p. 33). The cited implementers' report documents P2900R8, not P2900R14; records complementary gaps in the two compilers; and states that "The code has not been merged into any official branch" (p. 5). [45] The Clang status page recorded P2900R14 as not implemented at both cutoffs. [46] Between the cutoffs, on 2026-01-28, the base implementation of P2900R14 was merged into the GCC master branch. [47] When the March PDF was supplied, the prediction had come true for GCC, and the qualification "with upstreaming in progress" was out of date in its authors' disfavor, the GCC upstreaming having completed eight weeks earlier. Both implementations were publicly available on Compiler Explorer at both cutoffs, [48] and the response states the remaining gap itself: "While it has not been deployed to production, neither has any other major language feature adopted by C++ in any previous or current Standard" (p. 33).

"This concern repeats earlier objections ([P3173R0], [P3506R0], [P3573R0]) already considered repeatedly in EWG. No new information has been presented since." The same response states: "Deployment experience with the entire feature set of P2900 is admittedly still limited, in particular with pre and post" (P3846R1). [1] In the reflector discussion, such deployment was called valuable and then declined as a prerequisite, on the ground that it is not a reasonable expectation for a new language feature and that no comparable bar has been applied to any other language feature previously standardized (SG15 post 2909). [49]

The response admits the requested evidence is limited; the Discussion Status and the reflector answer rest instead on repetition, the feature's novelty, and the reasonableness of the evidentiary bar.

Verdict. Overall Support: Substantially Supported. Unsupported Subclaim: Yes, "fully implemented in two major compilers," as applied to P2900R14. Settled: No, replied.

5.3 Ten responses where support and weakness are both material

Concern 1: Safety and non-ignorable checks. The objection is that contract assertions make C++ less safe because they can be switched off. The response's strongest claim: "The ability to configure their evaluation semantics externally is a prerequisite for widespread adoption, not a defect" (p. 6), grounded in adoption history "as proven by decades of successful use of C assert" (p. 7). The narrower claim has documented support: production settings where observing or disabling assertions reduces outage risk, [50] and production, gaming, low-latency, server, and high-performance-computing environments with conflicting enforcement needs. [51] The categorical prerequisite claim has none: no study, survey, or usage data links the ignore mechanism to the adoption, and historical coexistence does not establish causation. The Rust comparison available to the objecting side is bounded in both directions: Rust checks a fixed class of operations and permits source-level bypass, and the 2021 study of restoring elided checks found little, no, or negative benefit in 76.4 percent of tested benchmarks and meaningful gains in 23.6 percent. [52] The November 2025 Android report describing checked-by-default Rust at scale [53] postdates the printed date and qualifies only the March PDF.

The reflector discussion accepted that the objection describes a real consequence of the proposal, agreed that the committee should confirm this is the intended outcome before standardizing or else reverse course, and wished for hard data. After describing the discussion as subjective, it assigned the burden to the prior vote: Overturning consensus of that strength should normally require significant new information, which the discussion was not seen to supply (SG15 post 2786). [54] The companion paper assigns non-ignorable checks to a later cycle: "[T]his extension is realistically too nontrivial to be approved in the C++26 timeframe and will have to target C++29." (P3500R1) [51]

The reflector answer acknowledges the consequence and the missing data before making consensus the reason not to revisit it; the companion paper answers the requested C++26 capability with future work.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, the categorical prerequisite claim. Settled: No, replied.

Concern 2: Cross-translation-unit semantics. The objection is that P2900 provides no consistent semantics across translation units compiled with different evaluation semantics. The response is a five-item strategy list requiring no change to P2900's specification, with the naive strategy's worst case scoped and qualified: "The worst case (barring compiler bugs such as those described in Concern 4) is that a contract assertion intended to be checked is instead ignored, which is no worse than if contract assertions did not exist" (p. 10). The qualified sentence is accurate as written, and the naive strategy is implemented in both compilers, [45] [55] but no public artifact exists for the two claimed prototype compiler implementations. The response reports deferred selection as "prototyped in GCC" with link-time optimization that "has been shown to work reliably in the GCC prototype" (p. 10). GCC's complete contract option table contains no link-time, load-time, or runtime selection option at either cutoff, [55] [56] and the initial upstream submission of the GCC implementation, posted to gcc-patches on 2025-11-17, describes "a command-line flag that determines a TU-wide contract evaluation semantic" and adds outlined checks and caller-side checks as its only extensions. [103]

The claim's provenance is a reflector thread six weeks before the printed date. One GCC implementer stated there that a proof of concept for link-time selection existed, with a demonstration that link-time optimization could yield good results, and that an attempt would be made to make it public as soon as possible. Two other participants in the same thread stated that no implementation allowing the semantic to be chosen at link time existed, and a later message described the prototype as known to its writer by the implementer's description. [101] [102] No public artifact followed by either cutoff. A GCC bug report filed 2026-02-13, between the two cutoffs, is titled on the premise that no "mechanism for runtime-configurable evaluation semantics" had been implemented, and the implementer's replies in it cite none. [104] A web and GitHub search for contracts-ABI implementations locates two public repositories by one Clang implementer, both specifying the violation entry-point ABI; the first, frozen since 2025-06-27, states that implementations "only support compile-time evaluation semantics," and neither describes or implements deferred selection. [57] [105] The second, created 2025-12-15, includes an Itanium ABI patch and may be the public form of the ABI proof of concept the response says "has been shared with WG21" (p. 10); if so, it surfaced between the cutoffs, and the March PDF did not cite it.

The authors' own rationale fixes the baseline. On 2025-03-14, Section 2.10 of P2899R1 stated of both compilers [20]:

> *Neither implementation currently does anything to influence the chosen version of an inline function with multiple definitions, so any non-inlined evaluation of such a function will have one of the possible behaviors (though which is unspecified and depends on the linker).*

The same paper adds in Section 3.5.5:

> *At the moment, implementations of Contracts in both GCC and Clang support per-translation-unit configuration of contract semantics... Work is in progress to add support for choosing caller-side evaluation in GCC.*

Caller-side evaluation is not deferred selection, so any deferred-selection prototype postdates March 2025, and none appears in the public record by either cutoff. After both cutoffs, on 2026-07-15, a configuration paper by a P3846R1 author and the GCC implementer reported "[d]ynamic selection of the evaluation semantic at link or run time via a named function" as partially implemented in branches of GCC and Clang, while stating that experimental implementations "express semantics at the TU, or higher, level" and describing the recovery of unchecked-build performance through link-time optimization prospectively, as something the implementation "will be able to" do. [106] The mechanism is described there as a configured selector function, a successor to or refinement of the `__current_semantic` emitter the response describes, and no measurement of the "reliably" claim has been published. Under the rule of Section 1, that paper changes the state of the objection for a future revision and does not make the printed sentence supported when written. The response states the residual gap itself: on the naive implementation, "users who do not fully control their build environment cannot reliably predict which evaluation semantic applies to non-inlined calls to `f`" (p. 10), and the strategy taxonomy records that mixed modes can reduce the minimum evaluation count to zero. [58]

The reflector discussion agreed that the problem exists and that nobody dismisses its existence. The same message then introduced the P3846 option set as the only three ways to address the problem, the first being that the committee accept the problem and standardize P2900 anyway, on the ground that it provides value to users despite the problem (SG15 post 2782). [59]

The rationale paper records the same acceptance eight months before P3846R1's printed date, as a trilemma: "satisfying all three simultaneously turns out to be impossible in this specific case. A conforming implementation of [P2900R14] inevitably needs to give up on one of the following items" (deterministic behavior in mixed mode, zero overhead on ignored assertions, or compatibility with current linkers) and adds that the problem "is unrelated to Contracts, has existed before Contracts, and is not made worse by Contracts" (P2899R1, Section 3.5.11). [20]

The reflector's first option and the rationale paper's trilemma both accept the problem rather than remove its mechanism, and together they are direct evidence for the difference between acknowledging an objection and resolving it. P3846R1's five-strategy list does not say which leg of the authors' own trilemma each strategy gives up; by that analysis, the deferred-selection and link-time strategies give up linker compatibility or zero overhead.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, the two claimed prototype implementations, for which no public record was located by either cutoff, and the first of which its implementer described on the reflector, six weeks before the printed date, as not yet public, per the search scope of Section 3. Settled: No, replied.

Concern 3: Dependency management. The objection, raised by P3849R0, is that contracts "introduce several new build configurations, but we have not yet seen concrete examples of how they interact with real-world build systems or complex dependency graphs." [5] The response answers with a replacement argument ("P2900 introduces no new configuration dimension," p. 11) and an example: "Boost.Build already added such support on top of the available GCC and Clang implementations of P2900. Adding this support took less than an hour of implementation effort" (p. 12). The example is genuine: The commit contains 149 additions across 9 files with a correct flag mapping, [60] and its example repository demonstrates the per-translation-unit model across the ignore-and-enforce combinations for static linking, with the shared-library case commented out from file creation. [61] The response describes the support as including "documentation covering scenarios such as static and dynamic linking" (p. 12); the runnable example exercises static linking only. The elapsed-time figure is an unwitnessed self-report, the commit author being both the B2 maintainer and a P3846R1 coauthor. It is excluded as evidence in either direction: It cannot establish that the integration was easy, and its exclusion does not establish that the integration was hard. The commit is public and stands on its own.

Neither the implementation-freedom argument nor the Boost.Build example establishes that contracts interact safely with complex dependency graphs. The package-manager case depends on a mechanism the example does not exercise: The rationale paper lists as the first use case for implementation-defined selection "[p]ackage managers and other forms of packaged software, which should not be forced to distribute a different binary for each possible evaluation semantic," resting on "the requirement that a conforming Contracts implementation should be allowed to enable and disable contract checks without requiring a rebuild," and records in the same section that only per-translation-unit configuration existed in either compiler (Section 3.5.5). [20] The B2 commit demonstrates per-translation-unit flag mapping, the one mechanism that does not deliver the package-manager benefit; the link-time, load-time, and runtime selection that would deliver it is the unimplemented mechanism of Concern 2.

The status is procedural: "These topics were explored in [P3321R0] and discussed by SG15 in Wrocław, with no concerns raised by that group. No new information has been presented since" (P3846R1). [1] The earlier analysis states the mechanism directly: "A Contracts design that requires build-time decisions regarding whether contracts are evaluated and what the consequences of contract violation are creates a significant burden for package managers" (P2877R0). [50]

The earlier paper argued for the per-translation-unit model to reduce that burden, and P3846R1 relies on that design plus the Boost.Build example. The reduction the earlier paper sought requires selection without a rebuild, which the authors' March 2025 rationale records as not yet implemented. For the broader request for complex dependency-graph evidence, the Discussion Status cites subgroup silence and the absence of new information.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, the documentation claim covering static and dynamic linking, where the runnable example covers static linking only. Settled: No, replied.

Concern 6: Implementation-defined behavior. The objection is that too much of P2900 is implementation-defined. The response explains the platform-dependence rationale and enumerates: "P2900 introduces exactly five implementation-defined properties" (p. 17). Four of the five (the termination mode, the replaceability of the violation handler, the choice of evaluation semantic, and the maximum number of repeated evaluations) map to entries in the incorporated working draft's own index of implementation-defined behavior. The fifth, "the exact behaviour of the default contract-violation handler" (p. 17), has no index entry because the draft governs it by recommended practice rather than by the words "implementation-defined" ([basic.contract.handler]/2); the property is real, and the draft's vocabulary is the reason it is not indexed. The same index lists seven contract-related entries, adding three the response omits: the virtual-destructor choice for `contract_violation`, the `comment()` contents, and the `location()` value. The two lists together name eight distinct properties, and the index was public 233 days before the printed date. [62] Two sources document the omission between them. P3321R0, described in the same paragraph of P3846R1 as discussing "the full list of implementation-defined behaviours" (p. 17), contains a section on the two string-valued entries. [63] The third omitted property, whether `contract_violation` is polymorphic, appears in the authors' own rationale paper, dated 2025-03-14, which lists six implementation-defined properties: the five P3846R1 names and "Whether the contract_violation object is polymorphic" (Section 2.10). [20] Each omission is therefore documented, one of them by the response's own authors.

After citing the prior response, EWG discussion, SG15 discussion, and the absence of new information, the response concludes: "None of these implementation-defined behaviours alter the way contract assertions are written nor do any represent an unresolved design gap" (P3846R1). [1]

The no-gap conclusion rests on an inventory the authors' own rationale paper had already counted differently. The platform-dependence rationale remains substantive, and the inventory error does not by itself establish a design gap.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, "exactly five," contradicted by the authors' own rationale and by the draft the response invokes. Settled: No, replied.

Concern 11: Predicate exceptions. The objection, recorded from FI-071, is that no implementation or deployment experience exists for non-Itanium ABIs and that Microsoft considers treating predicate exceptions as contract violations infeasible (p. 25). The response describes the two constituencies its design serves and states: "The approach in P2900 is the only known solution that satisfies both groups" (p. 25), with "[t]he overwhelming majority of predicates are trivially non-throwing" (p. 25) offered

without grounding; the response's "our experience" hedge attaches to a different sentence, that there is "no overhead for predicates that are not potentially throwing, which in our experience is the vast majority" (p. 25). The mechanism is coherent on its own terms. [43] The two documented alternatives satisfy only one constituency each: P3626R0, a coauthor's wording diff that P2899R1 records as supplementary material for EWG's Hagenberg discussion, [10] [20] and P3909R0, which notes non-Itanium translation costs without specifying a complete alternative. [64] On the platform the objection named, the response supplies no implementation or measurement, and none was located in the public record.

The procedural record shows division rather than resolution. SG21 adopted the design on 2023-05-18 (SF 8, F 7, N 2, A 0, SA 1). Three months earlier, two Issaquah polls had found no consensus either to treat predicate exceptions as violations (SF 7, F 7, N 3, A 3, SA 7) or to propagate them (SF 5, F 5, N 4, A 3, SA 10). [20] The 2023 decision adopted a design; it supplies no evidence about a platform, which is what FI-071's 2025 objection asks for. EWG at Hagenberg then polled unconditional unwinding of predicate exceptions at SF 12, F 18, N 11, A 15, SA 7, "No consensus for change." [40] [20] Of sixty-three votes cast, thirty favored the change the objection sought, twenty-two opposed it, and eleven were neutral. The twenty-two sufficed to deny consensus for change, and the thirty are sustained opposition to the design as shipped under any reading of the Directives' consensus definition. The authors' own rationale applies that term to the same body of objection, describing the pre-Hagenberg concerns as "restatements of known, sustained opposition to the Contracts design" (Section 2.6). [20] A poll records division; the record contains no reconciliation attempt after the vote.

> *The same position from Microsoft was raised previously in [P3506R0], addressed in [P3591R0], and given due consideration by EWG in Hagenberg. No new information has been presented since. (P3846R1)* [1]

This is the sole Not reached entry. Its Discussion Status consists entirely of a prior author response, committee consideration, and novelty. The rationale paper shows the authors were aware of that ABI's callee-side parameter destruction when specifying where postconditions must be checked (Section 3.5.1) and says nothing about predicate exceptions on it (Section 3.5.6). [20]

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, the trivially non-throwing proportion. Settled: No, not reached. The response supplies a coherent mechanism for its stated constituencies and no evidence on the interface where infeasibility was reported.

Concern 12: Static analysis. The objection is that P2900 does not support static analysis. The response argues that declaration-level syntax removes the macro limitations [65] and claims vendor activity: "Some static analysis providers (such as CodeQL) are already actively pursuing support for P2900 contract assertions in their tools" (p. 26), with the CppCon work combining "the CodeQL static analyser with the Z3 constraint solver to validate a wide range of contracts" (p. 27). The cited context paper, written by the talk's CodeQL copresenter, states that the prototype targets traditional assertions rather than P2900 contract specifiers, warns against using it to judge the overall feasibility of P2900 static analysis, and records that "the portions of this talk presented by GitHub are not an endorsement of P2900." [66] The prototype repository supports annotated assertion macros and BDE's `BSLS_ASSERT`; P2900 contract specifiers are a stated future hope: "In the future, we hope to support C++26 contract specifiers `pre(...)` and `post(...)`." [67] The syntax-level advantages are acknowledged from both directions. [65] [66]

The Discussion Status cites a paper chain and EWG discussion: "These concerns were raised in [P3362R0] and responded to in [P3376R0] and [P3386R1]. All three papers were discussed by EWG in Wrocław. No new information has been presented since" (P3846R1). [1] The reflector discussion stated the tradeoff more directly, characterizing the inability to check all contract assertions as a feature rather than a defect and as the route to runtime checks now and progressively more capable tools later (SG15 post 2991). [68]

The immediate mechanism is runtime checking and readable syntax; the broader tooling answer remains prospective.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, "validate a wide range of contracts," which outruns the demonstrated prototype. Settled: No, replied.

Concern 13: Complexity. The objection is that P2900 is too complex. The response reports that complete implementations "were produced relatively quickly by a tiny team" and states: "The implementers reported that P2900 is orders of magnitude simpler to support than modules, concepts, reflection, or even lambdas" (p. 27). The tractability half is documented: The cited implementers' report calls the specification "clear and implementable," [45] and a P2900R14 coauthor's later assessment calls the minimal form "fairly simple to implement." [69] The comparative magnitude is not: The cited report contains no comparison to the named features, and no measurement or attributed quotation exists. [45]

"The concern regarding complexity in [P3829R0] mirrors that of [P3573R0], which was discussed by EWG in Hagenberg. No new information has been presented since." The response then offers the adoption result as evidence: "[T]he strong plenary consensus in Hagenberg to include P2900 in the C++26 working draft is further evidence that its value is worth the cost" (P3846R1). [1]

The first statement cites prior discussion and novelty; the second offers the vote as evidence of value; neither documents the comparative magnitude.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, "orders of magnitude simpler." Settled: No, replied.

Concern 15: Future features. The objection is that adopting P2900 now forecloses or complicates future features, principally deep const. The response quotes SD-4's rule against delaying concrete proposals for hypothetical alternatives [70] and states: "Yet in more than four decades of C++ evolution, no proposal for deep const has ever been brought forward, and it appears doubtful that one will ever materialise" (p. 31). The historical sentence is false: P1974R0 proposes `propconst`, a language-level deep-const qualifier, [71] and P2670R1 revises that design line. [72] The falsity is narrow: P1974R0 predates P2900 const-ification, and no concrete compatibility analysis between P2900 and a complete deep-const design was located on either side.

> *Integration with future features was discussed extensively during P2900's development. Requirements were analysed in [P2885R3]; the possibility of deep const was examined in [P3261R2] and rejected via poll in both SG21 and EWG. No new information has been presented since. (P3846R1)* [1]

The cited polls retained P2900's constification design; they did not reject a complete deep-const proposal or establish compatibility with one. Discussion, polls on the present design, and novelty stand in for the requested compatibility analysis.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, the four-decades claim, which is false. Settled: No, replied.

Concern 16: Decomposition. The objection is that contract assertions could be composed from more primitive features standardized individually. The response identifies concrete omissions (a shared global handler, control over check injection, and the assertion marker that tools consume) in the proposed decomposition and states: "The idea to redesign contract assertions as a composition of more primitive features was first proposed in [P1893R0] and subsequently shown to be inadequate for the real-world use cases for contract assertions ([P1995R1])" (p. 32). The attached citation does not support the sentence: P1995R1 catalogs and polls use cases and does not mention or evaluate P1893R0. [73] [74] The identified omissions in the sketch stand unanswered, and a P2900R14 coauthor who did not sign P3846R1 argues separately that an atomic assertion marker supports tooling. [75]

> *Similar decompositions were proposed to SG21 by [P1893R0] and/or suggested as ideas in EWG but achieved no consensus as the basis for a proposal that meets the use cases P2900 was pursuing. Relaxation of the ODR was polled by EWG in Hagenberg, with consensus against. The specific decomposition proposed in [P3829R0] has not been explicitly discussed in WG21; otherwise, no new information has been presented since Hagenberg. (P3846R1)* [1]

The Discussion Status records the actual decomposition as not discussed and rests nevertheless on a poll on one component and the absence of other new information. The authors' own rationale describes that poll as "deliberately vaguely worded," taken "to gauge the interest of the room to 'do something about it'" (Section 3.5.11). [20]

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, "shown to be inadequate," by citation mismatch. Settled: No, replied.

Concern 18: Library hardening. The objection is that standard-library hardening cannot depend on contract assertions. The response names the four national body comments seeking decoupling in its first sentence (p. 35), explains that hardening can be specified through the contract-violation model without the literal syntax (p. 36), and states that "Both the libc++ and libstdc++ implementation currently being planned once contracts are available are conforming implementations of C++26 standard-library hardening on top of P2900" (p. 35). Six of its coauthors record the conforming macro approach both libraries use. [76] The response also anticipated the committee's later direction, calling the restriction of hardening to the enforce and quick-enforce semantics "a sound decision that the committee could make" (p. 36). Between the cutoffs, the committee made it: P3878R1 was adopted at Kona in November 2025 by unanimous consent in a motion stating that the change "addresses ballot comments RU-016, FR-001-014, FR-010-113, US 3-015, and US 61-112." [8] [77] The March PDF did not report the compromise adopted more than four months before it was built, and the phrase "on top of P2900" remains ambiguous between the literal syntax and the contract-violation model. The libc++ half of "currently being planned" is consistent with a coauthor's first-person knowledge; no public record of the claimed libstdc++ plan was located at either cutoff. The macro approach the response's own cited source records for both libraries is not an implementation on top of P2900. [76]

The response concedes that "the specification of standard library hardening, as it stands now, cannot be implemented purely in terms of the basic feature set in C++26 Contracts," then says that implementation strategies above P2900 are sufficient

(P3846R1). [1] The rationale states the missing mechanism more directly: "[W]e do not yet have the tools to distinguish contract assertions that should be treated differently in code," and leaves the assertion form to library implementers (P2899R1). [20]

By the response's own account the needed control is absent from the C++26 facility, and the closure rests on implementation discretion; P3878R1 later restricted which semantics qualify as standard-library hardening.

Verdict. Overall Support: Mixed. Unsupported Subclaim: Yes, the claimed libstdc++ plan, for which no public record was located. Settled: No, replied.

5.4 The fifteen Unsupported Subclaims

Because overall support is rated separately, each defect changes only the flag: Five of the fifteen responses remain Substantially Supported and ten remain Mixed, and no flag by itself invalidates the complete response that contains it.

Table 2. The fifteen material Unsupported Subclaims in P3846R1, in concern order. Each row quotes the subclaim with its page in P3846R1, states what the public record establishes, and classifies the defect.

Concern	Unsupported Subclaim	What the Record Establishes	Defect Kind
1	External configurability is "a prerequisite for widespread adoption, not a defect" (p. 6).	Configurability reduces adoption barriers in the environments that asked for it; no study, survey, or usage data links the mechanism to the adoption.	Unsourced causal claim
2	Deferred selection "prototyped in GCC"; link-time optimization "shown to work reliably in the GCC prototype" (p. 10).	No public code, measurement, or build log was located by either cutoff; GCC's contract option table contains no deferred-selection option at either cutoff, and the upstream GCC submission of 2025-11-17 describes a TU-wide flag only; the authors' March 2025 rationale records per-translation-unit configuration only, with caller-side evaluation the work in progress; the implementer stated on the reflector six weeks before the printed date that the proof of concept was not yet public; a GCC bug report of 2026-02-13 is premised on no runtime-configurable mechanism existing; the first public artifact is dated 2026-07-15, after both cutoffs, and describes the mechanism as a configured selector function	Unverifiable existence claim
3	Support includes "documentation covering scenarios such as static and dynamic linking" (p. 12).	The runnable example covers static linking only; the shared-library case is commented out from file creation.	Claim outrunning its evidence
4	Performance concerns are "equally unfounded" (p. 15).	The cited bug contains the GCC maintainer's report of a 47 percent gain on one workload and "quite considerable" surrendered opportunity in the narrow case.	False statement
6	"P2900 introduces exactly five implementation-defined properties" (p. 17).	The authors' own rationale paper lists six, adding the polymorphic `contract_violation` choice; the working draft's index lists seven contract-related entries and was made public 233 days before the printed date.	False statement
7	Destructive side effects from predicates "are rarely an issue" (p. 19).	No frequency data, survey, or defect study; teaching history establishes that the rule is taught, not how often the bug occurs	Unsourced quantitative claim

Concern	Unsupported Subclaim	What the Record Establishes	Defect Kind
9	Similar mechanisms are "widely and successfully used in major frameworks such as Qt and in game engines" (pp. 22-23).	No deployments and no outcomes are named for either ecosystem.	Unsourced empirical claim
11	"The overwhelming majority of predicates are trivially non-throwing" (p. 25).	Offered without grounding; the response's "our experience" hedge attaches to a different sentence, and no dataset is cited for either	Unsourced quantitative claim
12	The CppCon work combined CodeQL with Z3 "to validate a wide range of contracts" (p. 27).	The prototype validates annotated assertion macros and BDE's `BSLS_ASSERT`; P2900 specifier support is stated as future hope.	Claim outrunning its evidence
13	"The implementers reported that P2900 is orders of magnitude simpler to support than modules, concepts, reflection, or even lambdas" (p. 27).	The cited report contains no such comparison; no measurement or attributed quotation exists.	Unsourced quantitative claim
14	"[N]o proposals that included them gained consensus in EWG" (p. 28).	P3097R0 gained EWG consensus at St. Louis, a history the same section records one page later.	False statement
15	"[N]o proposal for deep const has ever been brought forward" (p. 31).	P1974R0 proposes `propconst`, a language-level deep-const qualifier, and P2670R1 revises that design line.	False statement
16	The decomposition idea was "subsequently shown to be inadequate" (p. 32), citing P1995R1.	P1995R1 catalogs and polls use cases and does not mention or evaluate P1893R0.	Citation mismatch
Concern	**Unsupported Subclaim**	**What the Record Establishes**	**Defect Kind**
17	"P2900 has been fully implemented in two major	The cited report documents P2900R8 with complementary gaps and code merged into no official	Claim outrunning

Concern	Unsupported Subclaim	What the Record Establishes	Defect Kind
	compilers" (p. 33).	branch; upstream Clang recorded P2900R14 as not implemented at both cutoffs.	its evidence
18	"the libc++ and libstdc++ implementation currently being planned once contracts are available" (p. 35)	No public record of the claimed libstdc++ plan was located at either cutoff; the cited source records the macro approach both libraries use, which is not an implementation on top of P2900.	Unverifiable existence claim

The defects fall into five kinds: four false statements (Concerns 4, 6, 14, 15), five unsourced claims of magnitude, frequency, cause, or deployment success (Concerns 1, 7, 9, 11, 13), two unverifiable existence claims (Concerns 2 and 18), one citation mismatch (Concern 16), and three claims that outrun their evidence (Concerns 3, 12, 17). The three responses without a flag (Concerns 5, 8, and 10) were tested against the same rule, and no material Unsupported Subclaim was found.

5.5 What would have settled each objection

A finding that sixteen objections were not settled invites the question whether settlement was reachable. Table 3 answers it one objection at a time. Each row names the record that would have moved the objection to Settled under Section 2's three forms: evidence the authors could have supplied (Form 2), a decision only the committee could take and record with reasons (Form 3), or a draft change the objector confirmed (Form 1). The two Settled rows are included to show the pattern the others lack. Thirteen rows name evidence the authors could have supplied themselves; two name a decision only the committee could take; one names a settlement that already existed on the record when the March PDF was built and went unreported.

Table 3. What would have settled each of the eighteen objections, with the form of settlement under Section 2. Where two routes are listed, either would have sufficed.

Concern	What would have settled it	Form
1. Safety and non-ignorable checks	A recorded EWG decision, with reasons, that externally configurable evaluation semantics are the intended design, which is the confirmation the reflector discussion itself said the committee should make before standardizing [54]; or adoption data linking the ignore mechanism to adoption.	3 or 2
2. Cross-translation-unit semantics	A public artifact of the deferred-selection prototype with a measurement behind "shown to work reliably"; or a recorded decision, with reasons, accepting the first leg of the authors' own trilemma. [20]	2 or 3
3. Dependency management	A demonstration of semantic selection without a rebuild in a package-manager or shared-library scenario, the case the Boost.Build example leaves commented out [61]; or a recorded SG15 finding with reasons.	2 or 3
4. One Definition Rule	A recorded EWG decision on the mixed-mode question in words, with reasons, in place of the poll its proposers describe as "deliberately vaguely worded." [20]	3
5. Modules	A specification or implementation of who controls the evaluation semantic across a module boundary; or a recorded decision deferring the question with an owner and a date.	2 or 3
6. Implementation-defined behavior	The complete inventory, eight properties rather than five, each with the reason it is acceptable; or a recorded decision with reasons that the inventory is acceptable as it stands.	2 or 3
7. Uncheckable guidance	Frequency evidence for "rarely an issue": a defect study, a survey, or an attributed count from a deployed codebase such as BDE; or the claim withdrawn and a recorded decision, with reasons, that the risk is accepted.	2 or 3
8. Constification	Settled. A mechanism with a stated purpose, its tradeoffs weighed on the record, migration evidence, and seven recorded decisions declining removal. [20] [37] [38]	2 and 3
9. Global violation handler	Named deployments and outcomes for the Qt and game-engine claim; or the claim dropped, the BDE history left as the evidence, [41] and a recorded decision on local handlers with reasons.	2 or 3
10. Consecutive assertions	Settled. A working mitigation for the canonical case, a counterexample against the alternative, and a recorded SG21 decision declining it. [26] [27]	2 and 3

Concern	What would have settled it	Form
11. Predicate exceptions	An implementation or measurement on a non-Itanium ABI, the interface where FI-071 reported infeasibility; or the Hagenberg poll converted into a recorded finding, in words and with reasons, that the objection is not persuasive. [40]	2 or 3
12. Static analysis	An analyzer that consumes P2900 contract specifiers, which the cited prototype states as a future hope [67]; or a recorded decision with reasons that the syntax-level argument suffices for C++26.	2 or 3
13. Complexity	The implementers' comparison attributed, quoted, and given its basis; or the plenary vote entered as a recorded finding, with reasons, that the cost is acceptable.	2 or 3
14. Missing features	A recorded decision, with reasons, that the MVP boundary is the intended C++26 scope, with each deferred feature given an owner and a date. The categorical consensus sentence corrected is a Table 2 repair, not a settlement.	3
15. Future features	A compatibility analysis between P2900 const-ification and the deep-const design line of P1974R0 and P2670R1 [71] [72]; or a recorded decision with reasons that no such analysis is required before adoption.	2 or 3
16. Decomposition	An evaluation of P3829R0's actual decomposition against the use cases, the analysis the P1995R1 citation was said to contain [73]; or a recorded decision on that decomposition with reasons.	2 or 3
17. Deployment experience	A deployment report covering the P2900R14 feature set, including pre and post, which the response itself says remains limited; or the reflector's argument that no comparable bar has applied to any prior feature [49] entered as a recorded finding with reasons.	2 or 3
18. Library hardening	Already on the record: P3878R1 modified the draft, and its authors record the commenters' confirmation. [77] A response that cited that change would have been settled by Form 1. The March PDF, built four months after Kona, did not report it.	1

Table 3 is the work list the settlement finding produces. It is also the answer to the structural objection Section 8 takes up: a rubric under which no objection could be settled would have no rows in this table, and every row names something specific. A later revision that closes a row by Form 2 changes that row's Settled entry and nothing else; the reconciliation record Section 9 describes is the committee's to make, and Section 10 finds it where it finds it however many rows the authors close.

6. What P3846R1's Own Text Records

Section 5 tests each response against the outside record, one concern at a time. The items below rest on P3846R1's own text and no external evidence, so a delegate who accepts none of the outside sources in Table 2 and rejects the rule of Section 3 entirely can still check every one with P3846R1 alone.

The categorical sentence that its own section falsifies. Concern 14 states that "no proposals that included them gained consensus in EWG" (p. 28) and records one page later that preconditions and postconditions on virtual functions "have been reviewed and approved with strong consensus in EWG" (pp. 29-30). The St. Louis poll the second sentence describes is the proposal the first sentence says does not exist.

The strongest sentence that its own page qualifies. Concern 17 states that "P2900 has been fully implemented in two major compilers" (p. 33) and eight lines later describes the same implementations as "nearly complete" with "upstreaming in progress" (p. 33). The cited implementers' report documents an earlier revision with complementary gaps in the two compilers and code merged into no official branch. [45]

The gap the response names itself. Concern 2 concedes that on the naive implementation, "users who do not fully control their build environment cannot reliably predict which evaluation semantic applies to non-inlined calls to `f`" (p. 10). The four strategies offered beyond the naive one rest on prototypes no public record contained by either cutoff.

The claim stated only conditionally. Concern 5 opens "In principle, inline functions in a BMI could carry additional information" (p. 16) and closes with modules as "only a partial solution" (p. 16). The response is accurate; the conditionality is why the objection was not settled.

The concern the revision history adds. P3846R0 [78] did not address the national body comments asking to decouple library hardening from contracts; P3878R0 recorded that "P3846 doesn't even mention these NB comments, and doesn't address them." [7] P3846R1's revision history records the repair: "Added Concern 18 about standard-library hardening in response to [FR-001-014], [US 3-015], [US 61-112] [FR-010-113], and [P3878R0]." The March artifact then did not report the compromise the committee adopted at Kona four months before the artifact was built. [8] [77]

The deferral pattern. Where the responses look forward, they point to work that does not yet exist: the labels proposal "currently being pursued for C++29," [76] the re-addition path for virtual functions, and the future linker support of Concern 2. The deliverables the ballot process expects are a disposition and revised text, [16] and a promise of future work is neither; a later, post-cutoff objecting paper states the same principle: "An objection answered by a promise of future work is answered only when that work arrives." [79]

All three of Section 2's forms occur within this dispute, and two of the eighteen responses meet one of them. Concern 10 supplied a working mechanism and a recorded SG21 decision declining the alternative. Concern 8 supplied a mechanism with its tradeoffs weighed on the record and seven recorded decisions declining removal. The same dispute produced a resolution by change: P3878R1 modified the draft, and its authors record that "the submitters of NB comments about standard library hardening confirmed that this change resolves their concerns." [77] Fifteen of the eighteen responses stop at explanation, one does not reach the objection's mechanism, and P3846R1's own text records where each stops.

7. In Their Own Words

The statements below show, across the response corpus and the reflector discussion around it, the pattern the concern-level quotations show locally: The authors invoked P3846R1 and its predecessors as the place where objections had already been considered and answered, and invoked repetition, absence of new information, roadmap choices, or a prior consensus as reasons to retain the C++26 design unchanged. They do not establish that any author consciously intended to substitute response for resolution.

7.1 P3846 as the answer

P3846R1 defines its two relevant fields this way [1]:

> *Discussion Status: Whether, when, and/or where this concern was previously considered, and whether we believe any new information has been presented that was not part of earlier consideration*
>
> *Response: A brief explanation of why the concern, when previously considered, did not prevent contract assertions from achieving consensus*

The test asks whether the issue was considered, whether information is new, and why the issue did not prevent consensus. It does not ask whether the objection's mechanism was resolved. The introduction states the result in the past tense: The paper describes "how they have been previously considered, analysed, and addressed" and "clarifies why they fail to justify removing contract assertions" from C++26. [1]

The reflector exchange over non-ignorable checks used that function directly, answering a live objection with the statement that the authors had done their best to explain the point in P3846R0, Concern 1. [54] Cited only as later corroboration, the post-cutoff C++29 roadmap preserves the same bibliography, listing the C++26 objections under "concerns" and P3846R0 among the "responses." [80] The timing of the response corpus is documented separately: P4195R0's Appendix A records that each response paper appeared after the poll whose outcome it defended and before the meeting at which that outcome would next be challenged, so that the room received objection and response together. [109]

7.2 Acknowledgment without resolution

The clearest statement is the reflector answer to the cross-translation-unit problem, which agreed that the problem exists and that nobody dismisses its existence. [59] That answer attributes the problem to the C++ compilation model, argues that any feature whose semantics change at build time encounters it, and sets out, as described in P3846, the only three ways it sees to address the problem [59]:

1. Accept the problem and standardize P2900 regardless, on the ground that it delivers value to users despite the defect.
2. Require deterministic semantics across translation units, which would stop the feature working with existing toolchains and, on this reasoning, prevent adoption.
3. Decline to standardize P2900 or any comparable feature.

The first listed answer accepts the problem and standardizes the design unchanged. A second reflector post described the same residual outcome and placed the controls outside the working draft. [81] On that account, the only negative possibility is that whoever builds the program believes a function was built one way when a different translation unit built it another way, so the semantic from that other translation unit applies. This is presented as the outcome current linker technology already permits, to be met in the future by better education about assuming control one does not have and by further options supplied in tooling outside the working draft while remaining fully conforming with C++26 Contracts.

The same post also points to linker technology available today, says vendors may reject or warn about mixed configurations, and says implementations need not support mixed mode. Those controls are permitted rather than required by the working draft. The post also states its affirmative case, which is recorded here so that the paraphrase is complete: Compilers' interprocedural optimization had begun turning previously benign ODR violations involving assertion macros into sources of undefined behavior, a language feature must fix that, and C++26 Contracts do, by specifying that mixed-mode evaluation of the same function is not undefined. [81] That is the response's claimed resolution, and it resolves the undefined-behavior consequence of mixing. The objection is the unpredictability of which semantic applies, and that is what the post's "only negative possibility" sentence leaves standing. In P3846R1 itself, the inability to predict which semantic applies remains stated as a fact, while the Discussion Status says the concern was addressed in a previous author response and presents no new information. [1]

7.3 Polls and consensus as disposition

P3591R0 described its purpose in terms stronger than response [43]:

> *Readers of those papers might mistakenly conclude that these concerns are new and profound flaws in the proposed Contracts facility and have not been previously discussed and addressed. In this paper, we present the missing background and facts related to each of the concerns raised; we describe how all the concerns have been discussed, how consensus was reached on their solutions, and why [P2900R13] is more than ready to be included in the C++ Standard.*

The rationale later described the treatment of strict predicates [20]:

> *At this point, the topic of 'strict contracts' was closed as far as SG21 was concerned. However, when the Contracts proposal was forwarded to EWG for the WG21 meeting in March 2024 in Tokyo, the author of [P2680R1] published [P3173R0], repeating the concerns and targeting EWG directly. EWG showed significant interest in pursuing the general concern raised.*

The first group considered the topic closed; the next group showed significant interest in the same concern. The same rationale names its own category for the papers that followed: "A comprehensive response to all these concerns was published in [P3591R0]." [20] In October 2025, the reflector discussion proposed an evidentiary rule. Having wished for hard

data to inform the question and having described the discussion as entirely subjective, it placed the onus on those seeking to overturn the Hagenberg consensus for including contract assertions as designed in C++26, recorded there as 100 in favor and 14 against, and held that overturning consensus of that strength should normally require significant new information, which the discussion was not seen to supply. [54]

The rule advanced is that the prior vote should shift the evidentiary burden. That is a defensible rule for reconsideration; it is not evidence that the acknowledged consequence was resolved.

7.4 Roadmap, mandate, and timing

For missing features, P3846R1 begins with the roadmap [1]:

> *The lifetime of P2900 really began when SG21 agreed to follow the path in [P2695R1] to pursue a specific plan to gain consensus on a minimal viable product (MVP) for contracts in C++. Starting with an MVP enables us to provide an already-standardised foundation on which consensus can be built for higher-level features that come later.*

For interaction with future features, the paper invokes a procedural bar [1]:

> *Procedurally, delaying the standardisation of P2900 due [to] the concerns of [P3829R0] about interactions with hypothetical proposals would be against WG21 practice.*

The reflector discussion applied the same division between MVP and later work to source-level semantic control, noting that SG21 and EWG had spent considerable time polling which features to target in the initial MVP and which to leave until later, and that the Contracts feature in the working draft is the result of such decision-making. [44]

The roadmap and polls explain why the feature set has its present boundary. They do not supply the capabilities assigned to the later layer or the compatibility analysis assigned to a future design.

7.5 Future work as closure

P3846R1's conclusion places present and future addressing side by side: "a description of how each concern is addressed in C++26" and "pointers to future (in-progress) proposals that will expand the use cases covered and further address any remaining concerns." [1] It then concludes [1]:

> *We hope that the detailed analysis provided herein demonstrates that the latest objections are neither new nor indications of fundamental flaws in the design of P2900. This design has already achieved strong consensus within WG21 and deserves to remain in C++26 as one of its cornerstone features.*

The same corpus records what the future work must provide. Non-ignorable checks were too late for C++26 [51]; syntactic control was "explicitly left to future proposals" [43]; local handlers were a post-C++26 extension [1]; and deployment with the complete feature was admitted to remain limited. [1] These statements can justify incremental delivery; they cannot also establish that the objections to the absent capabilities were resolved in C++26.

7.6 Later confirmation

The authors' C++29 roadmap, published after both scoring cutoffs, alters no rating. It corroborates the distinction between an existing response record and substantive work that remained [80]:

> *For concerns raised during the C++26 standardisation phase, see [P3173R0], [P3478R0], [P3506R0], [P3573R0], [P3829R0], [P3835R0], [P3849R0], [P3851R0], [P3911R2], [P4044R0]; for responses, see [P3500R1], [P3591R0], [P3846R0], [P3912R0], [P3946R0].*

The same paper states [80]:

> *The feature set included in C++26 provides a solid foundation, but lacks a number of extensions needed to support additional use cases for which there is already a demonstrated need. In particular, further work is required to make contract assertions truly usable at scale: in very large codebases, across libraries developed and distributed independently, and in specialised environments such as safety-critical systems. This is by design.*
>
> *Our other goal for C++29 is to address most, if not all, concerns that were raised repeatedly during the C++26 standardisation phase.*

P3850R1 categorizes P3846 as a response, while the repeated concerns and the capabilities needed for use at scale remain assigned to C++29. The response record existed, and the work needed to meet the concerns continued.

---

8. Expected Objections

This section states three objections to the assessment in their strongest form and answers them. The first is structural. Almost any response to a design objection leaves a material technical issue open, so a distribution of sixteen unsettled out of eighteen is a property of the rubric rather than a finding about P3846R1. The rubric never states what "Settled" would require. Objections that express priorities rather than technical questions cannot be resolved technically by construction. The second is P3846R1's own hedge. Its abstract qualifies the closure claim with "Some of the concerns are legitimate yet unavoidable with any viable assertion facility. Others reflect misunderstandings of the proposal, leading to inaccurate observations" (p. 1). On this reading, P3846R1 never claimed to resolve all eighteen, and measuring it against a resolution requirement would hold it to a duty the Directives do not impose. The third is the rationale paper. P2899R1 [20] provides, for each section of P2900R14, "a summary of the motivation for the proposed design, a history — as complete as possible — for the design decisions in that section," together with "polls that were taken and their results as called by the chair at the time" (Section 1). On this reading, Section 2's third form, a recorded decision with a stated rationale, already exists for every design decision the objections attack, and the present assessment cites P2899R1 eight times for tallies while denying P2899R1 the standing it claims for itself.

Section 2's three forms come from the ISO/IEC Directives and from disposition practice rather than from the three fields (Overall Support, Unsupported Subclaim, and Settled) of Section 3. A response resolves an objection by draft change with the commenter's confirmation, by explanation whose evidence suffices for its conclusion, or by recorded decision with a stated rationale; each is what a discharged attempt produces, and each leaves a record the objector can read. The definition of "Settled" in Section 3, that the supported response meets the stated objection on its own terms, is the same test stated for a single response. And the rubric does state what Settled would require, one objection at a time: Table 3 names it for all eighteen. Thirteen of the sixteen unsettled rows name evidence the authors could have supplied without a poll, which is Form 2 succeeding on its own; the objection that settlement was reachable only by vote is answered by the table.

The three forms are not theoretical; all three occur in this dispute. Concern 10 was answered by a mechanism plus a recorded decision. Concern 8 was answered by a mechanism whose tradeoffs were weighed on the record plus seven recorded decisions declining removal. The hardening concern was resolved by change: P3878R1 modified the draft, and its authors record the submitters' confirmation that the change resolves their concerns. [77] A property-of-the-rubric reading predicts that no response can qualify as Settled; the record contains two that are, and one draft change whose authors record resolution, against the same objections, in the same cycle.

Where an objection expresses a priority rather than a technical question, the third form is the disposition the Directives contemplate, and N4858's rejections supply examples. [14] For the fifteen, the record contains neither the decision nor the resolution.

P2899R1 does not supply Form 3, for four reasons.

1. **Authorship.** Form 3 is a committee act. N4858 is a disposition document prepared for the committee [14]; P2899R1 is a paper by four authors, among them all three authors of P2900R14, that, in its own words, "will largely defer to that paper" whose changes were adopted and "summarize" the decisions "along with polls that were taken" (Section 1). [20] P2899R1 is the proposers' account of the committee's reasons, and the Directives' vocabulary has a name for a

proposer's account: a response. That WG21 records tallies without rationales is a gap in committee practice rather than a fault of P2900R14's authors, who supplied the account the committee's own record lacks. The authors supplied replies; the process supplied no settlement. The first is to their credit and the second is the finding. It bears on this assessment only because Form 3 requires the committee's own decision, which no proposer can supply on the committee's behalf.

2. **Timing.** P2899R1 is dated 2025-03-14, and its poll record ends at Hagenberg. Of the five objecting papers P3846R1 answers, only P3506R0 predates it; P3835R0, P3829R0, P3849R0, P3878R0, and every national body comment on the committee draft came later. P2899R1 itself describes the concerns it does cover as "restatements of known, sustained opposition to the Contracts design" (Section 2.6), [20] which records that the opposition outlived the decisions it documents. This is a different point from the evidence cutoffs of Section 3: Form 3 requires a recorded decision on the objection, and a body cannot have recorded a decision on an objection that had not been raised when that body met. P2899R1 remains evidence of the design's rationale; it is not a disposition of the five objecting papers that came after it.
3. **Scope.** P2899R1 records decisions about the design. Form 3 requires a decision rejecting the objection. The two coincide only where a poll targeted the objection's own alternative, and where they do coincide, at Concerns 8 and 10, the responses receive credit. Where no poll addressed the objection's mechanism, P2899R1 supplies rationale for the design without disposing of the objection.
4. **The polls themselves.** P2899R1 characterizes two of the Hagenberg polls on which P3846R1's status entries rest. Of "P2900: add ODR to contracts," it states, "EWG took a deliberately vaguely worded poll to gauge the interest of the room to 'do something about it'" (Section 3.5.11); of "P2900: reduce the amount of Undefined Behavior in contracts," it says, "The group took a poll to do 'something' about this concern" (Section 3.6.1). [20] Those are the decisions behind Concerns 4, 7, and 16. A poll the proposers describe as deliberately vague, taken to gauge interest, is not a recorded decision with a stated rationale on any objection.

P3846R1 nowhere claims to have resolved the objections. Its closure claim is that they were addressed and extensively discussed, and the only place its text approaches the stronger word is a denial at Concern 6 that any of the implementation-defined properties represents an unresolved design gap. That is why the responses are measured against the attempt the Directives oblige rather than against a word the paper did not use.

Nor does the abstract's hedge exempt the closure claim. Its two classes are unnamed: No concern is mapped to "legitimate yet unavoidable" or to "misunderstanding," and no recorded decision disposes of any concern on either ground, so the hedge reserves an exemption without claiming it for anything. "Unavoidable with any viable assertion facility" is a universal claim about every viable facility, and the record contains a polled candidate counterexample: P3626R0, the alternative wording a coauthor prepared, which P2899R1 records as the wording diff supplied for EWG's Hagenberg discussion. [10] [20] The "misunderstandings" class has its own form under Section 2, resolution by explanation, and the fifteen flagged sentences of Table 2 are explanations whose evidence is inadequate for their conclusions; a misunderstanding is not corrected by an unsupported statement. "Extensively discussed" names the floor clause 2.6.5 sets and does not name the attempt at resolution the same clause obliges.

Candor is not penalized. The columns answer different questions by construction: Candor about limits is rewarded in the support column, where the bounded Concern 5 response is Supported with no flag, while the Settled column records the objection's state rather than the response's virtue. A candidly acknowledged open issue is still an open issue, and reclassifying it as closed would misstate the record, the error the support column checks for.

The boundary cases do not decide the finding. If every contested boundary moved one category in the authors' favor (Concern 11 to Replied and Concerns 5 and 18 to Settled), the distribution becomes four Settled and fourteen unsettled. A reviewer moving two categories, taking every Replied objection to Settled, would need to hold that fifteen responses each meet their objection on its own terms while fifteen of them also contain a material assertion the record does not support. That reading is available, and it is the point at which the disagreement becomes one about the evidence in Table 2 rather than about the boundaries. Short of it, the distribution is a property of the responses rather than of the instrument.

---

9. What a Discharged Duty Looks Like

This section states what a record supplying any of the three forms looks like; Section 10 asks whose duty it was to produce one. Nothing below is proposed; each element is in force today in a body operating under the same or stricter rules, and one of them is WG21's own.

The attempt at reconciliation is discharged when every sustained objector can find, on the record, the answer to their objection and can either accept that answer or name what the answer leaves open. That is ANSI's definition of a resolved comment, which turns on the commenter's acceptance of the resolution rather than on the outcome [18]; it is the IETF's rule that an issue may be dismissed with reasons but not ignored [17]; and it is Section 2's three forms stated for one objector at a time. Nothing in it entitles the objector to be accommodated, only to an answer that exists where they can read it.

The test has an empirical basis. In controlled dispute-resolution experiments, Thibaut and Walker found that participants judged a procedure fairer, and accepted an adverse decision more readily, when they controlled the presentation of their case, even where the decision itself rested with a third party. The effect was on acceptance of the outcome rather than on the outcome itself. [82] The bodies that record reconciliation wrote the same finding into procedure: The objector is heard on the record, told the answer, and given a path forward. An objector who disagrees with the outcome can still accept that the process ran. An objector whose objection was never answered has no such option, and in the record such objections return at the next stage.

Table 4. The record a discharged reconciliation duty produces and where each element is in force today.

The Record Shows	Where It Exists Today
Who sustained which objection, named per objection	ISO/IEC Directives 2.5.6: the leadership determines that opposition is sustained [13]
The chair's determination in words beside each tally	WG14 minutes label each straw poll a decision or an opinion and print the outcome in words [83]; WG21's Library Evolution poll-outcome papers print a finding under each tally and each voter's stated reasons beneath it [84]
Each author told one of three outcomes, with reasons	WG14's contributing page: after discussion "the committee will inform each author whether the committee has accepted the proposal as-is, requires a further revision of the paper, or rejects the proposal" [85]; WG14's issue procedure: "No issue should be resolved without some attempted form of communication with the original reporter" [86]
Each deferral with an owner and a date, then checked	RFC 2026: an Internet-Draft "that has remained unchanged in the Internet-Drafts directory for more than six months without being recommended by the IESG for publication as an RFC, is simply removed" [87]; an unowned promise lapses by default
An objection found unpersuasive only by recorded vote, with written reasons served on the objector	ASTM regulation 11.4.3.3: a motion to find a negative vote not persuasive "requires an affirmative vote of at least two thirds of the combined affirmative and negative votes cast," and "If the motion or ballot fails, the balloted item is removed from the ballot"; under 11.4.5, the reasons and the vote are recorded in the minutes; and under 12.8, the negative voter is notified of the disposition and the right to appeal [88]

The fifth row is where an adopted design comes back changed over its authors' objection, and WG21's own record shows it happening without any rule being added. In January 2022, the Library Evolution group polled forwarding P2300R4, `std::execution`, to the Library Working Group for C++23. The tally was SF 23, F 14, N 0, A 6, SA 11, a margin above two to one. [84] The recorded finding was no consensus. Its text names the operative objection, "sustained strong opposition against including such a large proposal into C++23 at such a late stage"; separates it from the design question, "The overall design still has strong support"; and records that the chair, a coauthor of P2300, "asked Vice Chairs Fabio Fracassi and Ben Craig to determine consensus on this poll" and "fully supports their decision." [84] Beneath the tally, the same document prints each voter's stated reasons, so the objection the finding names can be read in the objectors' own words. Four months later, the group forwarded P2300R5 for C++26 at SF 12, F 6, N 2, A 2, SA 3, with the recusal repeated. [89] The Strongly Against count had fallen from eleven to three. The design was not overruled. The objection was found persuasive on the record, the finding told the authors what had lost consensus and what had not, and the proposal returned on terms the objectors accepted.

The adoption arc contains the contrast case, and it should be stated in full because a WG21 reader will treat it as the committee's answer to the sustained opposition. At Hagenberg, after the concern-by-concern discussion, EWG polled "P2900: remove P2900 from CWG's consideration for C++26, find a different ship vehicle": SF 9, F 8, N 3, A 19, SA 41, "Consensus against." [40] The proposers' rationale describes the question as whether, "given the sustained opposition, the Contracts proposal should be withdrawn from consideration for C++26." [20] In one sense this is an answer: Sixty delegates were asked whether the opposition should stop the proposal and said no. The session notes record the tally and, as the next and final line of the session, the chair's congratulations to the authors. [107] The poll named no objection, stated no reason, and told no objector what the answer to their objection was. It decided the vehicle; it disposed of nothing in Table 1. It is also the event after which the objections went to plenary, to the committee-draft ballot, and to the DIS, which is what Table 5 records.

The two arcs can be set side by side. Table 5 places the Contracts record beside the P2300 record, stage by stage. One variable differs between the columns: whether anyone wrote down the answer.

Table 5. Two arcs of sustained opposition. The left column is the Contracts adoption arc as the proposers' rationale, the poll record, and the paper record describe it. The right column is the P2300 arc under the same rules and the same secretariat. Each row states what the record contained at that stage and what followed.

Stage	Contracts (P2900)	`std::execution` (P2300)
Opposition first sustained	SG21 considers strict predicates "closed as far as SG21 was concerned"; the objector republishes to EWG, which "showed significant interest." [20]	LEWG, January 2022: SF 23, F 14, N 0, A 6, SA 11, eleven Strongly Against on size and timing. [84]
First recorded decision	EWG Wrocław, November 2024: strict predicates SF 10, F 6, N 3, A 14, SA 16, recorded as "Consensus against, but P2900 would be in danger of failure in plenary" [20]; forwarding poll SF 25, F 17, N 0, A 3, SA 12, "consensus" [100]; a request from the floor to add context to the result statement, not acted on. [108]	Finding in words, "no consensus," naming the operative objection and separating it from the design; chair recused; each voter's reasons printed beneath the tally. [84]
Second recorded decision	EWG Hagenberg, February 2025: six polls, tallies with a one-phrase label; removal from C++26 consideration SF 9, F 8, N 3, A 19, SA 41, "Consensus against" [40]; "Congratulations to the authors." [107]	LEWG, May 2022: P2300R5 forwarded for C++26, SF 12, F 6, N 2, A 2, SA 3; recusal repeated. [89] Strongly Against fell from eleven to three.
Plenary	Hagenberg plenary adopts: 100 in favor, 14 opposed, 12 abstaining. [24] [110]	P2300 met new opposition on different grounds, complexity and teachability, at its 2024 plenary adoption; P4195R0 records that arc, and it is outside this paper's scope. [109] The two 2022 polls are cited for the form of the record and for the objection they resolved, not as a verdict on the design.
After adoption	Four objecting papers, P3829R0, P3835R0, P3849R0, P3878R0 [3] [4] [5] [7]; twenty-three national body comments on contract assertions at the committee draft. [8]	
Ballot	EWG Kona, November 2025: seven removal comments polled, each at ten to fourteen in favor against thirty-eight to forty-four Strongly Against, each labeled "No consensus for change" [110]; plenary rejects twenty-one of twenty-three comments, the vote the entire record [8]; P4238R1 and P4334R0 published for the DIS ballot [93] [79]; sixteen objections before the national bodies unsettled (Table 1).	

The finding in the right column was made for a large proposal, against a release deadline, over a numeric majority, with the chair holding a stake. Each of those conditions is present in the record Sections 5 through 8 assessed. The SG21 assistant chair is an author of P2900R14 and the first-listed author of P3846R1. [99] [1] [2] The SG21 chair is a coauthor of three of the five objecting papers, a stake on the opposite side that Section 10 states in full. [99] [3] [4] [7] The Wrocław forwarding poll recorded twelve votes at maximum opposition and zero neutrals. [100] The objections now before the national bodies are the ones the proposers' own rationale called sustained. [20]

None of the five rows requires ISO approval or a change to the Directives. The first is the Directives' own text. The second and third are the C committee's practice under the same secretariat, and the second is also WG21's own published practice in one subgroup. The fourth is the IETF's. The fifth is ASTM's, under ANSI accreditation, for decades. A record with these five parts lets a national body delegate answer, for each of the sixteen objections, where its answer is rather than whether the objection was heard.

---

10. The Duty to Reconcile Sat with the Convenership

Nothing in Sections 4 through 8 depends on this section. Where the duty to settle sat changes who owes the record; it does not change whether the record exists, and a reader who rejects every sentence below is left with Table 1, Table 2, and Table 3 unchanged.

One distinction governs this section: the difference between a party's act and the leadership's. P3591R0, P3846R1, and P2899R1 are the proposers' replies, and Section 4 credits them as such; the Directives' vocabulary for a proposer's account is a response. The record Table 4 describes is made by the chair or by the body, and no proposer can make it on the committee's behalf. The question below is therefore not whether the proposers answered, which Table 1 says they largely did, but whether the committee did. Why the committee's record takes the form it does, and what incentives that form creates for authors, reviewers, and chairs, is the subject of P4195R0 [109]; this paper takes the record as it finds it and asks whose office was to complete it.

Read against Table 4, the adoption arc's record is short in every row. Row 1: the determination that opposition was sustained was made, four times, in the proposers' rationale, and no record names which objector sustained which objection. [20] Row 2: the polls of record are tallies on the papers tracker with an outcome label at most and no stated reasons, [26] [39] [40] [42] and the proposers describe two of the load-bearing ones as polls to do "something" about a concern, one of them "deliberately vaguely worded." [20] The member-accessible EWG session notes for both meetings were examined for a finding the public record might lack. They record the discussion speaker by speaker. At Hagenberg the six tallies are entered at the end of the session with no finding in words, and the line that follows the removal poll is the chair's congratulations to the authors. [107] At Wrocław the notes record, after the strict-predicates poll, a request from the floor that more context be added to the result statement because a disputed consensus "shows up publicly without context," and a second participant's observation that declaring consensus is the chair's job rather than a matter of counting; the published result includes the label the rationale quotes and no further context. [108] [20] The request for the Row 2 record was made in the room and is itself in the notes. The Kona EWG session on the national body removal comments has the same form: the discussion speaker by speaker, seven

tallies each having the label "No consensus for change," and an eighth, on the Netherlands comment, labeled "Consensus to maintain contracts in the C++26 DIS"; the notes also record the chair asking whether a comment "will increase consensus" and directing presenters not to repeat what had been said that day, and a comment whose presenter was absent at its slot being marked heard until the convener asked that it be re-presented. [110] Row 3: "No new information has been presented since" closes sixteen of P3846R1's eighteen Discussion Status fields, fourteen of them without qualification and two naming a meeting; a seventeenth says "No other new information," which is counted separately because it concedes that some new information exists. [1] Row 4: the deferrals Section 6 lists have no owner and no date. Row 5: of the twenty-three national body comments on contract assertions at Kona, all but two were rejected, and the record of the rejections is the vote. [8] For the C++20 committee draft, WG21 published its disposition of comments as N4858, recording each rejection as "no consensus to adopt this change" with no further reason. [14] The WG21 document indexes for 2025 and 2026, through the August 2026 mailing, list no disposition of comments for the C++26 committee draft [22] [23]; if one is later published in the N4858 form, it will add the label and not the reason.

The sixteen open objections, therefore, raise a question of office rather than of authorship: Whose duty was the reconciliation that Section 2 requires? SD-4, WG21's own practices document, answers the structural question: It provides that "Subgroup chairs are appointed by the convener, and are selected to match the current needs of the subgroup. They have no fixed term"; that "The subgroup chair may take any polls they choose"; and that plenary consensus is "as determined by the Convener." [70] The ISO/IEC Directives assign the reconciliation duty to "the leadership": "If the leadership determines that there is a sustained opposition, it is required to try and resolve it in good faith." [13] The accountability provisions stop one level below the top of this chain: The term limits of clause 1.8.1 bind technical and subcommittee chairs but not working group convenors. [13] Within WG21, therefore, the duty and the discretion sit in a single office.

Herb Sutter held that office from 2002, when SC22 confirmed the appointment, [95] through the adoption arc assessed here [96] [8] and authored SD-4 itself, which the Direction Group's own reference list records as "[Sutter,2018] WG21 Practices and Procedures. ISO/IEC JTC1/SC22/WG21/SD-4. 2018-01-17." [97] Every chair who presided over the polls of record held office under his appointment, and SG21's is on the record: The convener announced in plenary that "John Spicer will be the chair." [98] Under those appointments, the SG21 consensus record documents the polls' tallies and no reconciliation process between them. [91] EWG's forwarding poll at Wrocław recorded a twelve-vote Strongly Against bloc with no recorded finding that the opposition had been answered. [100] Of the twenty-three national body comments on contract assertions at Kona, all but two were rejected. [8]

The SG21 chair's own position must be stated here, because it cuts toward the present author. John Spicer chaired SG21 from its 2019 chartering [98] through the Hagenberg adoption, [99] and he is a coauthor of three of the five objecting papers P3846R1 answers (P3835R0, P3829R0, and P3878R0), [3] [4] [7] of the hardening change adopted at Kona, [77] and of P4238R1 with the present author. [93] Two consequences follow, and both are accepted here. First, the finding of Table 4 is not suspended for SG21. The subgroup's polls of record are tallies with an outcome label, [26] [91] and no finding in words, no named sustainer, and no communication to an author is recorded beside any of them, under a chair who held the poll-taking discretion SD-4 grants. [70] The chair's own published account confirms the shortfall from inside the office: He records that participants "complained throughout the SG21 work that the pace was too fast," that he "heard those complaints" and "did not change course," and that he kept his technical objections to private conversations "until the paper was voted out of SG21 to EWG," which he now calls "a failure of chairing." [111] An objecting paper is an objection, not a reconciliation record, and

the chair's papers are recorded here as the former.

Second, the duty remains where the paragraphs above locate it. The dispositions P3846R1's Discussion Status fields invoke are, in the main, EWG polls at Wrocław and Hagenberg and the Hagenberg plenary vote; SG21 decisions are invoked in a minority of the fields, among them Concerns 8, 9, 10, and 16, and the two objections recorded here as Settled are among those. The sustained opposition the proposers' rationale names arose in EWG after the subgroup forwarded the design, and the five objecting papers and the national body comments were addressed to EWG, to plenary, and to the ballot rather than to SG21. A subgroup chair cannot reconcile objections lodged in a forum he does not chair, and SD-4 places both the appointment of that chair and the determination of plenary consensus with the convener. [70] The duty this section locates is the duty to answer the sixteen objections where they were raised. Where SG21's own record is short in the same rows, that shortfall is charged to the same chain, and no exemption is made here for the present author's coauthor.

The determination clause 2.5.6 requires is also on the record, in the proposers' own rationale, which uses the Directives' term four times: the pre-Hagenberg concerns were "restatements of known, sustained opposition to the Contracts design"; the safety claim was "another source of sustained opposition"; EWG polled "whether, given the sustained opposition, the Contracts proposal should be withdrawn from consideration for C++26" (SF 9, F 8, N 3, A 19, SA 41, consensus against); and after the Wrocław strict-predicates poll, "EWG had consensus against pursuing this direction, although sustained opposition to that decision remained," under a recorded result reading "Consensus against, but P2900 would be in danger of failure in plenary" (Sections 2.6 and 3.6.1). [20] The recorded answers to that determination were a removal poll, two gauge polls, one of which the same rationale calls "deliberately vaguely worded," and a response paper, P3591R0, which the rationale names as "a comprehensive response to all these concerns." [20]

The concentration of duty and discretion in one office is not itself the defect. A strong executive is how a volunteer committee breaks deadlocks, and the Directives vest the reconciliation duty in the leadership because a room of two hundred cannot reconcile anything. The defect is that the duty was not discharged on the record anywhere in the adoption arc, and it does not expire with the officeholder. Guy Davidson holds the convenership now: ISO selected him in November 2025, effective 2026-01-01, and the current revision of SD-4 names him as its reply-to. [90] [94] In February 2026, the Direction Group issued guidance on building consensus and converging proposals. [92] The sixteen open objections are now before the national bodies, the level where the resolution obligation governs outright.

---

11. Conclusion: Sixteen Objections Were Not Settled

Sixteen of the eighteen objections to C++26 contract assertions were not settled. That is not because the responses were poor: Assessed against the public record at two cutoff dates, they come out in their authors' favor on support (two Supported, six Substantially Supported, ten Mixed, none Unsupported or Contradicted), though fifteen contain a material assertion the record does not support and four contain one the record contradicts. It is because a reply is not a settlement, and the record contains replies. Measured against the resolution the ISO/IEC Directives oblige the process to attempt, for the objections P3846R1's abstract describes as addressed, two are Settled, fifteen are Replied without settlement, and one is Not reached. The responses produced substantive argument without closing the objections they answer, and that gap between the two axes is the finding. Whether the sixteen objections are correct is a question this paper did not ask; Table 3 states what would have settled each, and most rows name evidence rather than a vote.

The authors' own record shows how P3846R1 was invoked as closure despite that gap. Its Discussion Status asks whether a concern had been considered and whether information was new; its Response asks why the concern did not prevent consensus. In the reflector discussion, live objections were directed back to P3846, the cross-translation-unit problem was acknowledged, and acceptance of the problem, prior consensus, later tooling, roadmap boundaries, or future proposals were invoked as reasons to retain the C++26 design unchanged. The statements do not establish motive; they establish that the response corpus was invoked as a stopping rule even where its text recorded that the requested mechanism, evidence, or capability remained absent.

The same record contains points in P3846R1's favor. The Concern 2 worst-case sentence is scoped to the naive strategy and excludes the compiler-bug case by construction. The Concern 18 response names all four decoupling comments in its first sentence and anticipated the restriction the committee later adopted. Between the cutoffs, the base implementation of P2900R14 was merged into the GCC master branch, [47] a prediction the March artifact did not report in its authors' own favor. The two Settled responses show the obligation being met, and the hardening change shows the committee resolving an objection by draft change in the same dispute, on the change authors' record.

Four groups can build on this assessment. The convenership can set the record Section 9 describes beside the record Section 10 found, row by row. Authors preparing a future response paper can work from Table 2, which names the fifteen sentences to source, qualify, or remove, and from Table 3, which names what would settle each objection. The fields are independent by construction, so a Table 2 revision improves the responses without changing the Settled column. That column changes only when an objection is settled in one of the three forms: changed draft text with commenter confirmation, evidence that suffices for the stated conclusion, or a recorded decision with a stated rationale. Authors evaluating contract designs can build on the supported mechanisms: the bounded modules claim, the consecutive-assertion idiom, and the const-ification migration evidence. National body delegates weighing the C++26 draft can apply the Directives' obligation directly, because the objections assessed here include the comments before them.

Two findings follow from the record, and neither entails the other. The first concerns the objections: On the public record, sixteen of eighteen were not settled, resolved in none of the three forms. The second concerns the attempt the Directives oblige: No attempt at reconciliation is visible on the record. No entry names which objector sustained which objection, the polls of record have an outcome label at most and no stated reasons, the member-accessible session notes contain the discussion and no finding, sixteen of eighteen Discussion Status fields close with the observation that no new information has

been presented since, the deferrals have no owner and no date, twenty-one of twenty-three Kona comments were rejected with the vote as the entire record, and no disposition of comments for the committee draft has been published. The one recorded decision a reader might offer as the answer, the Hagenberg poll declining to remove P2900 from C++26, decided the vehicle and disposed of no objection, and Table 5 records what followed it. The first finding could be answered by settling the objections, and Table 3 says how, row by row. The second could be answered by leaving a record of the attempt, which the bodies surveyed in Section 9 already do. The first is within the authors' power and may change before the ballot closes; the second is the committee's, and no response paper can supply it. These sixteen objections are to the design the responses defend rather than to its C++26 vehicle, and they travel with any future vehicle for the same design until they are settled in one of the three forms.

Acknowledgments

The author thanks Mungo Gill, a coauthor of P3846R1, the paper assessed here, for independent verification of the quotations and citations, and Ville Voutilainen for a correction to Concern 18's quotation. Acknowledged individuals have not necessarily reviewed this paper and do not endorse its content.

Disclosure

The author provides information and serves at the pleasure of the committee. The author is president of the C++ Alliance and maintains coroutine-native I/O libraries under it. This paper was prepared with the assistance of generative tools; the author is responsible for its content.

This paper assesses P3846R1's eighteen responses against the public record at the two cutoffs stated in Section 3, a record broader than the sources P3846R1 itself cites. Its purpose is to put that assessment on the record where delegates weighing the C++26 draft can check it. It proposes no wording and requests no poll, and Section 9 describes practice in force in other bodies and in one WG21 subgroup while requesting none of it. Readers should nonetheless weigh the findings against the stake declared below.

The C++ Alliance has published a position, in P4238R1, [93] that the National Bodies vote No on the C++26 DIS ballot and return the draft over Contracts. This paper's findings support that position, and the author is a coauthor of P4238R1. This coauthorship is a material stake in the question under assessment. The author is likewise a coauthor of P4334R0, quoted in Section 6. One coauthor of P4238R1, John Spicer, chaired SG21 throughout the period assessed and coauthored three of the five objecting papers; Section 10 states what that fact does and does not change, and applies the paper's finding to SG21's record without exemption.

The ISO/IEC provisions cited in Section 2 bind the national body ballot and plenary level of the standards process rather than working group papers; Section 2 states the two reasons the resolution definition nonetheless supplies the test for P3846R1's closure claim. One further limitation of the method is the author's own: The assessment tests the sentences the author selected as load-bearing in each response, and a different selection could produce a different distribution of subclaim flags

even if every verdict here were upheld.

This paper asks for nothing. The record Section 9 describes can still be made at the ballot stage, before the national bodies vote; whether it is made is the convenership's choice, and this paper does not request it.

References

[1] P3846R1 - "C++26 Contract Assertions, Reasserted" (Timur Doumler, Joshua Berne, Gašper Ažman, Peter Bindels, Peter Dimov, Louis Dionne, Eric Fiselier, Mungo Gill, Pablo Halpern, Tom Honermann, Corentin Jabot, John Lakos, Nevin Liber, Lisa Lippincott, Ryan McDougall, Jason Merrill, Roger Orr, Nina Dinka Ranns, René Ferdinand Rivera Morell, Oliver Rosten, Iain Sandoe, Hui Xie, 2025).

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

[3] P3835R0 - "Contracts make C++ less safe - full stop" (John Spicer, Ville Voutilainen, José Daniel García Sánchez, 2025).

[4] P3829R0 - "Contracts do not belong in the language" (David Chisnall, John Spicer, Ville Voutilainen, Gabriel Dos Reis, José Daniel García Sánchez, 2025).

[5] P3849R0 - "SIS/TK611 considerations on Contract Assertions" (Harald Achitz, 2025).

[6] P3506R0 - "P2900 Is Still Not Ready for C++26" (Gabriel Dos Reis, 2025).

[7] P3878R0 - "C++26 Contracts are not a good fit for standard library hardening" (Ville Voutilainen, Jonathan Wakely, John Spicer, Stephan T. Lavavej, 2025).

[8] N5031 - "WG21 November 2025 Kona Hybrid meeting Minutes of Meeting" (Nina Dinka Ranns, 2025).

[9] LLVM Issue 28170 - "Calls to empty variadic functions in comdat no longer optimized out" (Warren Ristow, 2016).

[10] P3626R0 - "Make predicate exceptions propagate by default" (Timur Doumler, 2025).

[11] P2900R8 - "Contracts for C++" (Joshua Berne, Timur Doumler, Andrzej Krzemieński, 2024).

[12] P3097R0 - "Contracts for C++: Support for Virtual Functions" (Timur Doumler, Joshua Berne, Gašper Ažman, 2024).

[13] ISO/IEC Directives, Part 1 (consolidated) - Procedures for the technical work of ISO/IEC JTC 1, current consolidated edition; foreword principles and clauses 2.5.6 and 2.6.5, retrieved 2026-09-03.

[14] N4858 - "Disposition of Comments for CD Ballot, ISO/IEC CD 14882" (Barry Hedquist, 2020).

[15] ISO/TC 211 good practices, enquiry stage - ISO/TC 211 committee guidance on disposition of comments, retrieved 2026-09-03.

[16] N5028 - "C++26 CD summary of voting and comments" (Herb Sutter, 2025); the SC 22 summary of voting and collated comments on ISO/IEC CD 14882, circulated as a WG21 N-document.

[17] RFC 7282 - "On Consensus and Humming" (Pete Resnick, 2014); IETF consensus doctrine, retrieved 2026-09-03.

[18] ANSI Essential Requirements - "Essential Requirements: Due process requirements for American National Standards" (ANSI, 2025); the definition of a resolved comment in Annex A (Definitions), January 2025 edition, retrieved 2026-09-03.

[19] W3C Process Document - "W3C Process Document" (W3C); section 5.3, the formally-addressed adequacy test, retrieved 2026-09-03.

[20] P2899R1 - "Contracts for C++ - Rationale" (Joshua Berne, Timur Doumler, Rostislav Khlebnikov, Andrzej Krzemieński, 2025).

[21] cplusplus/papers issue 2455 - P3846 tracking issue, cplusplus/papers (2025).

[22] WG21 2026 paper index - Official WG21 paper index, 2026 directory, retrieved 2026-09-02.

[23] WG21 2025 paper index - Official WG21 paper index, 2025 directory, retrieved 2026-09-02.

[24] N5007 - "WG21 February 2025 Hybrid meeting Minutes of Meeting" (Nina Dinka Ranns, 2025).

[25] GCC commit 64674a2 - "c++, contracts: Allow contract checks as outlined functions." (Nina Ranns, Iain Sandoe, Ville Voutilainen, 2026).

[26] cplusplus/papers issue 2225 - SG21 poll on forwarding P3582R0, posted by the SG21 assistant chair (Timur Doumler, 2025).

[27] P3582R0 - "Observed a contract violation? Skip subsequent assertions!" (Andrzej Krzemieński, 2025).

[28] SG15 post 2744 - Reflector discussion of P3835 and P3846 Concern 10 (SG15 mailing list, 2025).

[29] LLVM commit 5ce3272 - "Don't IPO over functions that can be de-refined" (Sanjoy Das, 2016).

[30] GCC Bug 70018 - "[6 Regression] Possible issue around IPO and C++ comdats discovered as pure/const" (Sanjoy Das, 2016).

[31] GCC Bug 121936 - "[14/15/16/17 Regression] Invalid optimisation (at O3) based on bodies of vague linkage functions" (Iain Sandoe, 2025).

[32] GCC commit cac7958 - "c++, contracts: Work around GCC IPA bug, PR121936 by wrapping terminate." (Nina Ranns, Iain Sandoe, 2026).

[33] P3499R1 - "Exploring strict contract predicates" (Timur Doumler, Lisa Lippincott, Joshua Berne, 2025).

[34] PRE31-C. Avoid side effects in arguments to unsafe macros - SEI CERT C Coding Standard (Carnegie Mellon University Software Engineering Institute).

[35] SonarQube S3346 - "Expressions used in Debug.Assert should not produce side effects," SonarQube static-analysis rule for C#, added 2017-06-01 and first shipped in sonar-dotnet 6.0 (SonarSource, 2017); retrieved 2026-09-02.

[36] PVS-Studio V6055 - PVS-Studio static-analysis diagnostic for Java, retrieved 2026-09-02.

[37] P3336R0 - "Usage Experience for Contracts with BDE" (Joshua Berne, 2024).

[38] P3261R2 - "Revisiting const-ification in Contract Assertions" (Joshua Berne, 2024).

[39] cplusplus/papers issue 2062 - EWG Wrocław poll on removing const-ification, posted by the EWG chair (JF Bastien, 2024).

[40] cplusplus/papers issue 1648 - EWG Hagenberg contracts polls, posted by the EWG chair (JF Bastien, 2025).

[41] P2811R7 - "Contract-Violation Handlers" (Joshua Berne, 2023).

[42] cplusplus/papers issue 1822 - EWG St. Louis polls on P3097R0, posted by the EWG chair (JF Bastien, 2024).

[43] P3591R0 - "Contextualizing Contracts Concerns" (Joshua Berne, Timur Doumler, 2025).

[44] SG15 post 2980 - Reflector discussion of P3400, the Contracts MVP, and features left until later (SG15 mailing list, 2025).

[45] P3460R0 - "C++ Contracts Implementers Report" (Eric Fiselier, Nina Dinka Ranns, Iain Sandoe, 2024).

[46] Clang C++ support status - Clang C++ support status page, version at the 2025-11-03 cutoff; the version at the 2026-03-23 cutoff records the same status.

[47] GCC commit c928dc51 - "c++, contracts: C++26 base implementation as per P2900R14." (Iain Sandoe, 2026).

[48] Compiler Explorer C++ compiler configuration - compiler-explorer/compiler-explorer, `etc/config/c++.amazon.properties`; the contracts toolchains appear identically in the versions at both cutoffs.

[49] SG15 post 2909 - Reflector discussion of the deployment-experience threshold for Contracts (SG15 mailing list, 2025).

[50] P2877R0 - "Contract Build Modes, Semantics, and Implementation Strategies" (Joshua Berne, Tom Honermann, 2023).

[51] P3500R1 - "Are Contracts 'safe'?" (Timur Doumler, Gašper Ažman, Joshua Berne, Ryan McDougall, 2025).

[52] Safer at Any Speed: Automatic Context-Aware Safety Enhancement for Rust - Proceedings of the ACM on Programming Languages, Volume 5, Issue OOPSLA, Article 103 (Natalie Popescu, Ziyang Xu, Sotiris Apostolakis, David I. August, Amit Levy, 2021).

[53] Rust in Android: move fast and fix things - Google Security Blog (Jeff Vander Stoep, 2025).

[54] SG15 post 2786 - Reflector discussion invoking P3846R0, the Hagenberg vote, and the new-information threshold (SG15 mailing list, 2025).

[55] gcc/c-family/c.opt at 436aff90 - GCC contracts development fork (villevoutilainen/gcc), compiler option table at the branch head of 2025-10-19.

[56] gcc/c-family/c.opt at bd0dde45 - GCC master compiler option table at the last commit touching the file on or before the 2026-03-23 cutoff.

[57] efcs/contracts-abi - Contract-violation entrypoint ABI design document (Eric Fiselier, 2025); README and .gitignore only, frozen since 2025-06-27.

[58] P3267R1 - "C++ contracts implementation strategies" (Peter Bindels, Tom Honermann, 2024).

[59] SG15 post 2782 - Reflector discussion acknowledging cross-translation-unit unpredictability and presenting three responses (SG15 mailing list, 2025).

[60] Boost.Build commit 3b20a4e - "Add initial support for {CPP}-26 Contracts for GCC based toolsets (like clang)." (René Ferdinand Rivera Morell, 2025).

[61] grafikrobot/cpp_contracts_example - C++ Contracts example repository (René Ferdinand Rivera Morell, 2025).

[62] N5008 - "Working Draft, Programming Languages - C++" (Thomas Köppe, 2025).

[63] P3321R0 - "Contracts Interaction With Tooling" (Joshua Berne, 2024).

[64] P3909R0 - "Contracts should go into a White Paper - even at this late point" (Ville Voutilainen, 2025).

[65] P3386R1 - "Static Analysis of Contracts with P2900" (Joshua Berne, 2024).

[66] P3893R0 - "The CppCon 2025 Talk on Contracts and CodeQL in Context" (Mike Fairhurst, 2025).

[67] advanced-security/codeql-contracts-smt-z3 - SMT constraint solving in CodeQL with Z3; frozen since 2025-09-19.

[68] SG15 post 2991 - Reflector discussion of runtime checks and future static-analysis tooling (SG15 mailing list, 2025).

[69] P4020R0 - "Concerns about contract assertions" (Andrzej Krzemieński, 2026).

[70] SD-4 - "WG21 Practices and Procedures" (Herb Sutter, 2024). Revision of 2024-12-30, the text in force throughout the adoption arc assessed here.

[71] P1974R0 - "Non-transient constexpr allocation using propconst" (Jeff Snyder, Louis Dionne, Daveed Vandevoorde, 2020).

[72] P2670R1 - "Non-transient constexpr allocation" (Barry Revzin, 2023).

[73] P1995R1 - "Contracts - Use Cases" (Joshua Berne, Timur Doumler, Andrzej Krzemieński, Ryan McDougall, Herb Sutter, 2020).

[74] P1893R0 - "Proposal of Contract Primitives" (Andrew Tomazos, 2019).

[75] P3859R0 - "Assertions are not necessarily for changing program behavior" (Andrzej Krzemieński, 2025).

[76] P3912R0 - "Design considerations for always-enforced contract assertions" (Timur Doumler, Joshua Berne, Gašper Ažman, Oliver Rosten, Lisa Lippincott, Peter Bindels, 2025).

[77] P3878R1 - "Standard library hardening should not use the 'observe' semantic" (Ville Voutilainen, Jonathan Wakely, John Spicer, Stephan T. Lavavej, 2025).

[78] P3846R0 - "C++26 Contract Assertions, Reasserted" (Timur Doumler, Joshua Berne, et al., 2025).

[79] P4334R0 - "P2900 Contracts' fundamental flaws" (Bjarne Stroustrup, José Daniel García Sánchez, Vinnie Falco, John Spicer, Ville Voutilainen, 2026).

[80] P3850R1 - "A proposed plan for extending Contracts in C++29" (Timur Doumler, Joshua Berne, 2026); later corroboration only.

[81] SG15 post 2805 - Reflector discussion of mixed-translation-unit semantics and future tooling outside the working draft (SG15 mailing list, 2025).

[82] Thibaut, John W., and Laurens Walker - *Procedural Justice: A Psychological Analysis* (Hillsdale, NJ: Lawrence Erlbaum Associates, 1975). Experimental findings on process control and acceptance of adverse decisions.

[83] N3227 - "Draft Minutes for 22-26 January, 2024," WG14 meeting in Strasbourg (WG14, 2024); straw polls labeled "decision" or "opinion" with the outcome stated in words, retrieved 2026-09-03.

[84] P2459R0 - "2022 January Library Evolution Poll Outcomes" (Bryce Adelstein Lelbach, Fabio Fracassi, Ben Craig, 2022); Poll 1 on P2300R4, the recorded finding, and the voters' stated reasons.

[85] Contributing to WG14 - WG14 contributing page; the three outcomes communicated to each author after discussion, retrieved 2026-09-03.

[86] N3002 - "Issue Tracking Proposal for C" (Aaron Ballman, 2022); the reporter-communication requirement in Section 2.4.

[87] RFC 2026 - "The Internet Standards Process - Revision 3" (Scott Bradner, 1996); Section 2.2, the six-month expiration of Internet-Drafts, retrieved 2026-09-03.

[88] Regulations Governing ASTM Technical Committees - ASTM International; section 11.4.3.3 on the two-thirds not-persuasive motion, 11.4.5 on recording reasons and the vote in the minutes, and 12.8 on notifying negative voters of the disposition and the right to appeal, retrieved 2026-09-03.

[89] P2575R0 - "2022-05 Library Evolution Poll Outcomes" (Bryce Adelstein Lelbach, Fabio Fracassi, Ben Craig, Inbal Levi, 2022); Poll 2.1 on P2300R5.

[90] Trip report: November 2025 ISO C++ standards meeting (Kona, USA) - Herb Sutter's blog, 2025; the convenership succession announcement.

[91] P2521R4 - "Contract support - Record of SG21 consensus" (Andrzej Krzemieński, Gašper Ažman, Joshua Berne, Bronek Kozicki, Ryan McDougall, Caleb Sunstrum, 2023).

[92] P4024R0 - "Guidance on Building Consensus and Converging Proposals" (Jeff Garland, Paul E. McKenney, Roger Orr, Bjarne Stroustrup, David Vandevoorde, Michael Wong, 2026).

[93] P4238R1 - "Returning C++26 for the Evaluation It Skipped" (Vinnie Falco, Ville Voutilainen, José Daniel García Sánchez, John Spicer, 2026). Revision 1, in the same mailing as this paper; R0 stated the same position.

[94] SD-4 - "WG21 Practices and Procedures" (Guy Davidson, 2026). Current revision, dated 2026-05-11.

[95] N1409 - "Minutes of ISO WG21 meeting, October 20, 2002" (Robert Klarer, 2002); records "Plum noted that Herb Sutter was confirmed as convenor of WG21."

[96] N5005 - "WG21 2025-01 Hagenberg Admin telecon minutes" (Nina Dinka Ranns, 2025).

[97] P0939R0 - "Direction for ISO C++" (Howard Hinnant, Roger Orr, Bjarne Stroustrup, Daveed Vandevoorde, Michael Wong, 2018).

[98] N4826 - "WG21 2019-07 Cologne Minutes of Meeting" (Nina Dinka Ranns, 2019).

[99] isocpp.org committee page - "The C++ Standards Committee - ISOCPP" (isocpp.org, 2025); snapshot of 2025-06-13 recording "SG21, Contracts: John Spicer (Edison Design Group), assistant chair Timur Doumler."

[100] cplusplus/papers issue 1648 - EWG Wrocław poll forwarding P2900R11 to CWG and LEWG for C++26, posted by the EWG chair (JF Bastien, 2024).

[101] SG15 post 2605 - Reflector discussion of link-time selection of the evaluation semantic, including a GCC implementer's statement that a proof of concept existed and would be made public as soon as possible, and two statements that no such implementation existed (SG15 mailing list, 2025-09-24).

[102] SG15 post 2608 - Reflector discussion describing the link-time or runtime hook approach as "currently being prototyped in GCC"; the follow-up at SG15 post 2612 describes the prototype as known to the writer by the implementer's description (SG15 mailing list, 2025-09-26).

[103] gcc-patches, "[PATCH 0/9] C++26 Contracts initial implementation" - Cover letter of the initial upstream submission of the GCC contracts implementation (Iain Sandoe, 2025-11-17), read through the marc.info mirror; the cover letter lists outlined checks (8/9) and caller-side checks (9/9) as the only proposed GNU extensions, and the TU-wide flag description is in the design comment of patch 3/9.

[104] GCC Bug 124088 - "-fcontracts-conservative-ipa is unnecessary and wrong as long as no mechanism for runtime-configurable evaluation semantics is implemented" (Matthias Kretz, 2026-02-13), with replies by the GCC contracts implementer.

[105] efcs/contracts-abi-specification - Contract-violation entry-point ABI specification with runtime example and Itanium ABI patch submodule (Eric Fiselier, created 2025-12-15, last pushed 2026-02-23).

[106] P3595R0 - "Configuration of Contract Evaluation Semantics" (Joshua Berne, Iain Sandoe, 2026-07-15); later corroboration only.

[107] EWG Contracts session notes, Hagenberg - "Contracts in EWG," WG21 committee wiki, February 2025; member access. Records the concern-by-concern discussion, the six closing tallies, and the chair's closing line. The notes' tallies for three of the six polls differ from the chair-posted tracker record by a few votes each; the tracker [40] is treated as authoritative for tallies throughout, and the notes are cited for the discussion and for what follows the final poll. Retrieved 2026-09-19.

[108] EWG Contracts session notes, Wrocław - "NotesEWGContracts," WG21 committee wiki, November 2024, revision of 2024-11-22; member access. Records the discussion preceding the strict-predicates, const-ification, and forwarding polls, including the request from the floor to add context to the result statement. Retrieved 2026-09-19.

[109] P4195R0 - "WG21 Game Theory: The Culture That Emerges From SD-4" (Vinnie Falco, 2026). Companion paper in the same mailing; Section 1 states the reconciliation test as a property of the record, Section 5 records the P2300 adoption arc through 2024, and Appendix A records the timing of the P2900 response papers against the polls they defended.

[110] EWG NB comments session notes, Kona - "NB Comments," WG21 committee wiki, November 2025, revision of 2025-11-05; member access. Records the discussion of the seven national body comments seeking removal or a Technical

Specification and the Netherlands comment supporting retention, the eight tallies with their labels, and the Hagenberg plenary count of 100 in favor, 14 opposed, and 12 abstaining, attributed to N5007. Retrieved 2026-09-19.

[111] P4381R0 - "Chairing SG21 and then objecting to Contracts" (John Spicer, 2026). The SG21 chair's account of the pace, of complaints heard and not acted on, and of technical objections kept private until the design left the subgroup.