A Reader's Guide to the August 2026 Mailing

Document Number:	D4199R0
Date:	2026-08-31
Intent:	Inform
Audience:	WG21

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

Table of Contents

Abstract

Ten papers deliver a working profiles prototype covering seventy-seven cases of core-language undefined behavior with zero foundational wording changes, propose five governance correctives drawn verbatim from ISO Directives, and document how Contracts reached C++26 without a dedicated architecture ballot - backed by deployment evidence from eight shipping systems.

This paper summarizes 10 papers published in the August 2026 mailing. It is a reading guide: an executive summary that identifies the logical series within the collection, describes what each series delivers, and provides individual summaries of every paper. It asks for nothing.

1. Disclosure

The author provides information and serves at the pleasure of the committee.

This paper asks for nothing.

---

2. Executive Summary

P4317R1 defines `std::core_ub` under the P3589R2 Profiles framework - the same 77 runtime-checkable cases of core-language undefined behavior that P3100R8 enumerates, guarded by a single annotation, with zero foundational wording changes and no contract-violation handler dependency. A working Clang prototype demonstrates 7 locally checkable cases on Compiler Explorer. P4297R1 reconstructs the seven consensus polls that advanced P3100 and shows that not one of them adopted the architecture placing Profiles beneath its machinery. P4306R1 assembles more than a hundred cited sources and measures both candidate owners against four criteria already in the committee record. P4330R0 walks ten contracts papers line by line and catalogues the mechanisms by which a profiles safety framework loses its independence. Together these four papers establish that the alternative to waiting for contracts already exists, already works, and now runs live - the enumeration consensus in P3100R8 is separable from the architecture dispute over routing.

The compound finding across this cluster is architectural: the contracts substrate captures the violation response, the configuration design space, and the definition of class invariants - leaving profiles with no independent path to deployment. Each paper documents a different axis of that capture, and reading them together makes visible what no single paper shows: the foreclosure is not one decision but a distributed accumulation of incremental merges whose aggregate effect was never the subject of a standalone ballot. P4317R1's prototype demonstrates that separability is not a theoretical argument - it is running code.

Three papers converge on a single question - what happens when a runtime check fires - and the deployment record answers it uniformly. P4308R1 enumerates all eight possible responses to a throwing implicit contract assertion, expands EWG's current four-option menu, and scores them against six requirements drawn from P3100R8's own prose plus five dimensions from the public record. P4310R1 surveys every hardened implementation - libc++, libstdc++, MSVC STL, glibc, Google's server fleet, Android IntSan, UBSan, and three major framework assertion macros - and finds that not one defaults to continuation in production. P4318R1 applies net-present-value reasoning to the portable continuation guarantee and rejects it on two independent grounds: the marginal value is near zero because the capability already ships as vendor opt-ins, and the cost falls as a perpetuity on every conforming implementation while the benefit decays as codebases complete adoption windows.

The trilemma at the center of this cluster is irreducible: preserving the noexcept operator's value, its meaning, and stack unwinding cannot all hold at once. The deployment evidence settles what the theoretical argument leaves open - production code terminates or traps, and the two options that let the exception escape are not deployed anywhere.

Three papers address WG21's internal governance with the specificity of proposed text. P4130R0 identifies five passages where SD-4 diverges from the ISO/IEC Directives and proposes five drop-in replacements: fixed three-year chair terms with committee confirmation, reconciliation-based consensus, unrestricted national body ballot comments, no penalty for exercising the formal appeal process, and explicit polls for substantive questions. P4238R0 documents how Contracts

advanced through sixty-three papers in ten months without breaking a single procedural rule, and asks seven of twenty-six voting P-members to return the DIS for the evaluation it skipped. P4302R1 proposes that any single delegate may block a counted tally on an unmailed revision, confining a narrow exception to wording corrections and feature removal.

Read together, all ten papers constitute a single integrated argument at three levels. Any one paper gives the reader a specific tool: a governance amendment, a deployment survey, an evidence record, a working prototype. Any one cluster gives the reader a compound understanding - of architectural independence, of violation-response design, or of procedural dynamics - that no single paper achieves alone. The full collection gives the reader a qualitatively different picture: a demonstration that the committee's existing frameworks, existing deployment evidence, and existing procedural rules already supply every piece needed to ship runtime safety checking for C++29 without waiting for the substrate proposed to absorb it.

Three entry points serve different readers. An implementer or library maintainer should begin with P4317R1, which specifies std::core_ub with enough detail - and a working prototype - to evaluate cost, coverage, and interaction with existing sanitizer infrastructure. A language designer concerned with the noexcept semantic and the violation-response design space should begin with P4308R1, whose eight-option enumeration and requirements grid frame the tradeoff EWG must resolve. A national body delegate preparing a DIS ballot position should begin with P4238R0, which reconstructs the procedural sequence and states the specific ask - seven No votes out of twenty-six - with the supporting evidence cross-referenced to the companion governance papers.

---

3. Individual Papers

3.1. P4130R0 - SD-4: Five Correctives Inspired by ISO Directives

SD-4 lets one office appoint every subgroup chair, set the meeting schedule, and declare consensus - and nothing in the document checks any of those powers. P4130R0 identifies five specific passages where WG21's internal practices document diverges from the ISO/IEC Directives that govern it, and proposes five drop-in text replacements drawn directly from those Directives: fixed three-year chair terms with committee confirmation, reconciliation-based consensus in place of the two-to-one vote ratio, unrestricted national body ballot comments, no penalty for exercising the formal appeal process, and explicit polls for substantive questions instead of silence-as-agreement. Every corrective is a single passage swap, and none requires a poll, a study group, or a national body ballot to adopt.

3.2. P4238R0 - Returning C++26 for the Evaluation It Skipped

Every C++ DIS ballot from C++11 through C++23 passed unanimously; the C++ Alliance is asking National Bodies to break that streak by voting No on C++26. The paper documents how Contracts advanced through sixty-three papers in ten months, each merging an incremental change that became the settled baseline for the next poll, until the accumulation was treated as a mandate that no single vote had granted - all without breaking a single procedural rule. It reconstructs the closed TS path, the reversed burden of removal, the compressed review windows, and the contradictory consensus determinations, drawing on reflector statements from Stroustrup, Dos Reis, and co-author Spicer - who chaired SG21 during the period in question and now concludes that P2900 "does not make anything better." The authors request no floor time, no poll, and no committee action; they state a position and ask seven of twenty-six voting P-members to return the draft for the evaluation it skipped.

3.3. P4297R1 - Severing P3100's Profiles Claim from Its Case-by-Case Review

Seven consensus polls advanced P3100 into case-by-case EWG wording review, yet not one of them adopted the architecture that places Profiles beneath P3100's machinery - and by the time an adoption poll exists, seventy-seven individual case approvals will stand behind the claim. The paper reconstructs the full poll history (Table 1), identifies the six foundational wording clauses whose approval settles the architecture before any remaining case is reached (Table 2), and reports that no published WG21 paper contests the characterization through the 2026-05 mailing. It proposes three polls - a scope statement, a process commitment, and an evidence standard - that let the wording review proceed unblocked while requiring the layering question to face its own dedicated ballot.

3.4. P4302R1 - Any Delegate May Object to a Poll on an Unmailed Revision

WG21 has recorded counted polls on paper revisions that no national body expert outside the meeting room had seen - at Croydon, six of nineteen papers changed between the mailing and the vote. P4302R1 proposes an SD-4 amendment under which any single delegate may block a counted tally on an unmailed revision, keeping the result out of the minutes while preserving discussion and a qualitative record of sentiment. The paper documents cases from both Croydon and Brno where in-meeting revisions established committee precedent before the wider review chain received the text, and confines a narrow final-meeting exception to wording corrections and feature removal. R1 replaces R0's flat prohibition with an objection right - a lighter mechanism that preserves the principle that counted results should follow national body review.

3.5. P4306R1 - Configuring Runtime Checking: Profiles and Implicit Contract Assertions

Two proposals have converged on the same configuration mechanism for runtime checking of core-language undefined behavior, and P3100R8's own text states that if both are kept, one must be specified in terms of the other - but the committee has not decided which. P4306R1 assembles the public record across more than a hundred cited sources - vendor documentation, maintainer statements, deployment reports from Google, Apple, Mozilla, Android, and Chrome - and measures both candidate owners against four criteria already in the committee's record: deployment experience, existing practice, systematic coverage, and guarantee strength. The finding is that no criterion settles ownership: P3100 leads on systematic coverage, the named-guarantee form has a decade of production deployment across three vendors with measured cost, both proposals' specifications are unshipped, and existing practice reads both ways depending on which clause of P2000R5 one invokes. The paper is a companion to P4297R1 and supplies the evidence record an explicit EWG ballot would weigh.

3.6. P4308R1 - Eight Responses to a Throwing Implicit Contract Assertion

Of the eight ways a language can respond when an implicit contract assertion's violation handler throws, four are deployed in shipping code - and the two that let the exception escape are not deployed anywhere. P4308R1 enumerates the full response space for this question, expanding the four options currently before EWG (from P3100R8) to eight, and scores all of them against six requirements drawn from P3100R8's own prose plus five dimensions from the public record - deployment lineage, security posture, compatibility direction, diagnostics, and implementation experience. The requirements grid, built on Option A's own criteria, gives Option A genuine wins on unwinding and on adding no new semantics, but the deployment, security, and compatibility dimensions point toward the options that terminate or trap. The paper surfaces a trilemma at the center of the design - preserving the noexcept operator's value, its meaning, and stack unwinding cannot all hold at once - and names no winner, leaving EWG to choose which property to sacrifice.

3.7. P4310R1 - Hasta la Vista, Undefined Behavior: Why std::core_ub Should Terminate by Default

Every hardened implementation the authors surveyed - libc++, libstdc++, MSVC STL, glibc, Google's server fleet, Android IntSan, UBSan, and three major framework assertion macros - terminates or traps on a detected core-language violation in production; not one defaults to continuation. P4310R1 takes that deployment record and adds two load-bearing findings: the committee already decided the adjacent case when it adopted P3878R1 into C++26 for library hardening, and continuing past a detected violation executes user code on a state the language does not define - the exact hazard the security literature treats as worse than stopping. The paper argues that std::core_ub should reuse the C++26 enforce semantic, which preserves the handler invocation for telemetry while introducing no new semantic and leaving noexcept unchanged. It carves out two exceptions - the defined-replacement class (such as wrapped signed overflow) where the corrupted-state objection does not apply, and an explicit opt-in continuation mode for adoption periods - and makes no request of the committee.

3.8. P4317R1 - A Profile for Runtime-Checkable Core-Language Undefined Behavior: std::core_ub

The same 77 runtime-checkable cases of core-language undefined behavior that P3100R8 enumerates can be guarded by a single profile - with zero foundational wording changes and no contract-violation handler dependency. P4317R1 defines std::core_ub under the P3589R2 Profiles framework: one annotation ([[profiles::enforce(std::core_ub)]]) activates all checks, violations terminate, and 15 cases such as signed overflow and division by zero receive well-defined replacement values instead. A working Clang prototype demonstrates 7 locally checkable cases on Compiler Explorer, with live examples covering misaligned access, null dereference, out-of-bounds subscript, and arithmetic UB. Positioned as design exploration rather than a proposal for adoption, the paper serves as evidence that the enumeration consensus in P3100R8 is separable from the architecture dispute over routing.

3.9. P4318R1 - Transient Benefit, Perpetual Cost: Implicit Core-Language Assertions

P4318R1 applies net-present-value reasoning to a single standardization decision - whether to make the log-and-continue response for implicit contract assertions on core-language undefined behavior a portable guarantee every implementation must carry - and the arithmetic rejects it on two independent grounds. First, libc++ and Bloomberg's BDE already ship the same capability as vendor opt-ins documented for adoption-period use only, so the marginal value of the portable guarantee is near zero. Second, even granting a positive benefit, the benefit decays as codebases complete their adoption windows while the cost - implementer maintenance the libc++ team has stated (P3191R0) it will not carry, definitional coupling through a noexcept meaning-shift, and cognitive load - falls on every conforming implementation as a perpetuity. The paper explicitly credits P3100R8's enumeration of undefined behavior and its terminating responses as carrying durable value; the model prices only the one slice whose benefit is transient and whose delivery mechanism is the hardest to reverse.

3.10. P4330R0 - Analysis of Contracts Papers

Ten contracts papers are examined through two lenses and every one of them either subordinates profiles to the contracts substrate or moves program behavior out of the source and into the build system - often both. P4330R0 walks each proposal line by line, cataloguing the mechanisms by which a profiles safety framework loses its independence: the program-wide violation handler captures the response to core-language undefined behavior, Labels capture the configuration design space, class invariants are defined within contracts, and continuation past a detected violation becomes the default that termination must opt into. The analysis notes that hardened standard libraries already ship terminating runtime checks for bounds, null pointers, and iterator validity today - without P2900, without a program-wide handler, and without Labels - while the framework proposed to absorb their configuration exists only as a Compiler Explorer prototype. The paper asks for nothing; it documents the architectural foreclosure and leaves the committee to decide whether that foreclosure is acceptable.

---

4. Conclusion

This reading guide covers 10 papers from the August 2026 mailing. The author hopes it helps the reader find the papers most relevant to their work and interests.

References

[1] P4130R0 - "SD-4: Five Correctives Inspired by ISO Directives" (Vinnie Falco, 2026).

[2] P4238R0 - "Returning C++26 for the Evaluation It Skipped" (Vinnie Falco, 2026).

[3] P4297R1 - "Severing P3100's Profiles Claim from Its Case-by-Case Review" (Vinnie Falco, 2026).

[4] P4302R1 - "Any Delegate May Object to a Poll on an Unmailed Revision" (Vinnie Falco, 2026).

[5] P4306R1 - "Configuring Runtime Checking: Profiles and Implicit Contract Assertions" (Vinnie Falco, 2026).

[6] P4308R1 - "Eight Responses to a Throwing Implicit Contract Assertion" (Vinnie Falco, 2026).

[7] P4310R1 - "Hasta la Vista, Undefined Behavior: Why std::core_ub Should Terminate by Default" (Vinnie Falco, 2026).

[8] P4317R1 - "A Profile for Runtime-Checkable Core-Language Undefined Behavior: std::core_ub" (Vinnie Falco, 2026).

[9] P4318R1 - "Transient Benefit, Perpetual Cost: Implicit Core-Language Assertions" (Vinnie Falco, 2026).

[10] P4330R0 - "Analysis of Contracts Papers" (Vinnie Falco, 2026).