# A response to the Tokyo EWG polls on the Contracts MVP (P2900R6)

Timur Doumler (papers@timur.audio)
John Spicer (jhs@edg.com)

**Document #:** P3197R0
**Date:** 2024-04-12
**Project:** Programming Language C++
**Audience:** SG21, EWG

## Abstract

This paper summarises the actions that SG21 has decided to take in response to the EWG guidance polls taken at the March 2023 Tokyo meeting after the first round of EWG design review of the Contracts MVP paper [P2900R6].

## Introduction

In order to structure our responses to the individual polls, we classified the response to each EWG poll into one of the following three buckets:

1. There might be an issue in the MVP. More discussion in SG21 is needed.
2. SG21 is confident that what is in the MVP now is the correct solution, but we recognise that we need to do more work to persuade those with concerns in EWG.
3. No action from SG21 is required at this time.

## Responses to individual polls

### Poll 1

**P2900R6 Contracts should have *enforce* semantics only, and should not have *ignore* nor *observe*.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 6 | 1 | 3 | 15 | 24 |

**Response:**
Bucket 3: No action from SG21 is required at this time.

The poll result seems clear enough to satisfy the criteria for Consensus Against a change to the MVP. Disallowing the *ignore* and *observe* evaluation semantics would fail to meet important design requirements and make the MVP non-viable. We believe that there are ways to address the concerns with regards to safety-critical software raised in EWG without removing these evaluation semantics from the MVP.

## Poll 2

**P2900R6 Contracts in constant expressions should have *enforce* semantics only, and should not have *ignore* nor *observe*.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 7 | 10 | 16 | 7 | 8 |

**Response:**
Bucket 2: SG21 is confident that what is in the MVP now is the correct solution, but we recognise that we need to do more work to persuade those with concerns in EWG.

We will provide stronger motivation for the current specification and present this to EWG in St. Louis.

## Poll 3

**P2900R6 Contracts should specify contracts on virtual functions in its Minimal Viable Proposal.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 8 | 15 | 10 | 13 | 5 |

**Response:**
Bucket 1: There might be an issue in the MVP. More discussion in SG21 is needed.

We have several papers proposing support for virtual functions on the SG21 agenda: [D3097R0], [P3165R0], and [D3169R0] (the first two papers are proposing the same solution). We should continue discussing these papers in SG21. If we gain consensus on a solution, we should poll whether we want to merge that solution into the MVP, or treat it as a post-MVP feature. The EWG poll result suggests that the former might make the MVP more palatable to EWG; SG21 will take this into account.

## Poll 4

**P2900R6 Contracts should specify contracts on function pointers in its Minimal Viable Proposal.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 6 | 6 | 14 | 15 | 8 |

**Response:**
Bucket 3: No action from SG21 is required at this time.

There is currently no paper proposing a specification for how contracts on function pointers could work. In the absence of a paper, there is nothing actionable for SG21. This is a complex topic and any possible solution would have interesting and wide-reaching consequences. Several co-authors of P2900 have spent a significant amount of time thinking about this problem, but no concrete proposal for how contracts on function pointers could be specified emerged so far from these discussions. We believe that discussing this further before we have consensus on a solution for virtual function support is not helpful, as there might be a dependency given the similarity of the problem. We further believe that, considering the complexity of the issue and the EWG poll results, function pointer support does not seem essential for the MVP and could be added as a post-MVP extension.

## Poll 5

**P2900R6 Contracts should specify contracts on coroutines in its Minimal Viable Proposal.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 6 | 3 | 11 | 19 | 7 |

**Response:**
Bucket 3: No action from SG21 is required at this time.

A paper [P2957R1] proposing to reconsider coroutine support is in the SG21 queue. The author confirmed that this paper is not targeting the MVP but should instead be considered as a post-MVP extension.

## Poll 6

**P2900R6 Contracts should not allow throwing exceptions out of a violation handler.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 8 | 16 | 12 | 7 | 14 |

**Response:**
Bucket 1: There might be an issue in the MVP. More discussion in SG21 is needed.

SG21 recognises that throwing contract-violation handlers are a necessity for some important use cases, and therefore does not intend to remove them from the MVP. Simultaneously, SG21 recognises that there is sustained opposition to their existence in the language among some WG21 members due to ramifications for code that does not intend to use throwing handlers and for the wider `noexcept` policy.

There is a paper in the works illustrating the important use cases for throwing handlers which we aim to present to EWG in St. Louis. We further intend to discuss the wider philosophical

questions regarding throwing handlers in SG21 pre-St. Louis and investigate solutions other than outright removing throwing handlers that could address the current concerns. As part of this discussion, we will see [P2946R1], [P3101R0], [D3205R0], and possibly other papers in this space.

## Poll 7

**P2900R6 Contracts should not be able to evaluate preconditions/postconditions/assertions more than once per invocation.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 13 | 8 | 15 | 10 | 8 |

**Response:**
Bucket 1: There might be an issue in the MVP. More discussion in SG21 is needed.

The current specification in the MVP allows for an unbounded number of evaluations of a single contract assertion. Given the feedback we received in Tokyo and the above poll results, we no longer have consensus that this is the correct solution. Since the Tokyo meeting, multiple papers appeared addressing this issue, [P3119R0] and [D3228R0]. We will discuss these papers and possibly change the current specification in the MVP. We recognise that this is a case of conflicting design requirements, as disallowing more than one evaluation seems to preclude allowing both callee- and caller-side checking of precondition and postcondition assertions without an ABI break, which some members of SG21 and co-authors of P2900 consider a crucially important requirement.

## Poll 8

**P2900R6 Contracts should expose less undefined behaviour than regular C++ code does.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 17 | 12 | 12 | 12 | 5 |

**Response:**
Bucket 1: There might be an issue in the MVP. More discussion in SG21 is needed.

As discussed in Tokyo, there are two orthogonal cases how Contracts can expose undefined behaviour:

a) Undefined behaviour elsewhere can lead to elision of an *observed* contract assertion.
b) Undefined behaviour while evaluating the contract predicate itself can lead to elision of an *enforced* contract assertion, or any other unintended effects.

Case a) can be addressed by making a contract assertion an optimisation barrier. We currently do not have a specification tool for this in the language. We requested that EWG

reconsider [P1494R2], which proposes such a specification tool. Independently from EWG, SG21 will also consider this paper and discuss whether we want to integrate it into the MVP.

For case b), we currently do not have a specification paper that we could incorporate into the MVP to address this case; until such a specification paper is available, this case is not actionable for SG21. We have [P2680R1], which is merely a direction paper. We have discussed this paper extensively in SG21 and we do not see value in repeating this discussion in SG21 at this time. Instead, we have deferred this matter to SG23 (Safety and Security) and we will wait and see what guidance on this topic we receive from them.

## Poll 9

**P2900R6 Contracts - there should be some usage experience of contracts in an implementation of the STL (without necessarily having a paper to adopt these changes) before contracts can move to plenary.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 16 | 30 | 12 | 3 | 0 |

**Response:**
Bucket 3: No action from SG21 is required at this time.

Our understanding is that the authors of [P3191R0] intend to provide us with usage experience of Contracts in libc++. We will continue to engage with them and see what they come back with.

## Poll 10

**P2900R6 Contracts - there should be some usage specification of contracts in the STL before contracts can move to plenary.**

| SF | F | N | A | SA |
|---|---|---|---|---|
| 6 | 10 | 11 | 14 | 15 |

**Response:**
Bucket 2: SG21 is confident that what is in the MVP now is the correct solution, but we recognise that we need to do more work to persuade those with concerns in EWG.

We believe that if EWG wants to add precondition and postcondition specifiers to the *specification* of the STL (rather than merely gaining *implementation* experience), they should first consult with LEWG if this is a direction that LEWG supports. In addition, we would need a paper proposing concrete additions to the specification of the STL, and probably an LEWG policy paper as well. We are not aware of anyone planning to bring forth such papers. In addition, there are reasons why we believe that adding contract assertions to the *specification* of the STL is a bad idea.

However, we recognise that in order to alleviate concerns, it would be useful to present to EWG what contract assertions in the STL would actually look like if we *were* to specify them (without actually proposing such a specification). There is some material available on this matter ([P2755R1] section 4.4, [D3212R0]) which can be presented to EWG.

# Bibliography


[P1494R2] Davis Herring: "Partial program correctness". 2021-11-10
[P2680R1] Gabriel Dos Reis: "Contracts for C++: Prioritizing Safety". 2022-12-15
[P2751R0] Joshua Berne: "Evaluation of *Checked* Contract-Checking Annotations". 2023-01-14
[P2755R1] Joshua Berne, Jake Fevold, and John Lakos: "A Bold Plan for a Complete Contracts Facility". 2024-04-11
[P2756R0] Andrew Tomazos: "Proposal of Simple Contract Side Effect Semantics". 2022-12-31
[P2900R6] Joshua Berne, Timur Doumler, and Andrzej Krzemieński: "Contracts for C++". 2024-02-29
[P2946R1] Pablo Halpern: "A Flexible Solution to the Problems of `noexcept`". 2024-01-15
[P2957R1] Andrzej Krzemieński and Iain Sandoe: "Contracts on Coroutines". 2024-01-13
[D3097R0] Timur Doumler, Joshua Berne, and Gašper Ažman: "Contracts for C++: Support for Virtual Functions". 2024-03-10
[P3101R0] Ran Regev and Gašper Ažman: "Differentiating potentially throwing and nonthrowing violation handlers". 2024-01-22
[P3165R0] Ville Voutilainen: "Contracts on virtual functions for the Contracts MVP". 2024-02-26
[D3169R0] Jonas Persson: "Inherited Contracts". 2024-03-21
[P3119R0] Joshua Berne: "Tokyo Technical Fixes to Contracts". 2024-04-03
[P3191R0] Louis Dionne, Yeoul Na, and Konstantin Varlamov: "Feedback on the scalability of contract violation handlers in P2900". 2024-03-19
[D3205R0] Gašper Ažman, Jeff Snyder, and Andrei Zissu: "Throwing from a noexcept function should be a contract violation". 2024-04-02
[D3212R0] Andrzej Krzemieński: "The contract of sort()". 2024-03-29
[D3228R0] Timur Doumler: "Contracts for C++: Revisiting contract check elision and duplication". 2024-04-12