Project: ISO JTC1/SC22/WG21: Programming Language C++
Doc No: WG21 **P1823R0**
Date: 2019-07-18
Reply to: Nicolai Josuttis (nico@josuttis.de),
Ville Voutilainen (ville.voutilainen@gmail.com),
Roger Orr (rogero@howzatt.demon.co.uk),
Daveed Vandevoorde (daveed@edg.com),
John Spicer (jhs@edg.com),
Christopher Di Bella (cjdb.ns@gmail.com)
Audience: EWG
Prev. Version:

Remove Contracts from C++20

This is a proposal nobody likes and we are not happy to make it. But we have to suggest to remove Contracts from C++20, because the feature is not ready for standardization yet and to continue to force to have it in C++20 is taking the whole C++20 at risk.

Motivation

Contracts were added to the working draft in Rapperswil 2018 with the usual assumption that we have agreement on the general goal and programming API of contracts. It turned out that this is not the case.

In fact, we currently face the following situation:

- We had major design changes proposed and accepted by EWG this meeting (Cologne, July 2019)
- We have significant disagreement whether these changes make things better or worse. In fact, even the initial authors of Contracts for C++20 have no agreement.
- We still discuss what the design changes mean.
- We still design details on the fly.
- Some major concerns were raised regarding the proposed changes.
- We have no implementation experience (such as applying the currently proposed feature in the standard library)

In our train model, we do not require that features be in final form when they are adopted into the working draft under the assumption that they will be fixed as needed before the train departs. Part of the train model must be an agreement that if there is not consensus that a feature is in acceptable form when the train is about to depart that the feature must be taken off the train and that it has to catch the next train.

According to P1000, at this meeting we should finish C++20 wording, after being feature complete for one or two meetings. For contracts, we de-facto failed to follow this schedule.

This would only be acceptable, if we all agreed that it was sane to do so and still saw a clear path for a successful integration of Contracts in C++20. However, with the current disagreement keeping Contracts alive in C++20 takes the whole C++20 standard (with a huge amount of other useful and long awaited features) at significant risk.

And we should learn from other experiences in the past where we did regret rushing a feature being in such a state (e.g., `std::auto_ptr`, export templates, `std::async()`).

The situation simply is that contracts are obviously not ready for C++20.

For this reason it is no longer appropriate to make Contracts part of the C++20 standard.

But didn't we already reject a removal?

We had a similar proposal with **D1426R1 "Pull the Plug on Contracts?"**, which was discussed in Kona in February 2019.

This paper to remove contracts was rejected by the following vote:

P1426 Pull the Plug on Contracts?

SF	F	N	A	SA
2	7	4	14	7

However, the situation has changed significantly:

- In February we had some time until C++20 and thought issues will be (possible to be) sorted out.
- Now we don't have time and have major issues we know we can't solve in time.

So based on this new situation, we have to vote on this question again, assuming that a removal is the only sane option we have now.

Final Disclaimer

This paper doesn't in any way judge or compare the different proposals. All we say is that we are obviously not done yet.

We have removed features before (such as Concepts for C++11) and came back with better approaches. Let's do the same with Contracts.

Thanks to all who contributed with their time and hearts so far on this feature. This was not a waste of time. Good things take time!

Proposed Wording

(All against N4820)

In **[tab:lex.name.special]** remove `axiom` and `audit`.

In **[basic.def.odr]/12**, remove a bullet:

~~— if D invokes a function with a precondition, or is a function that contains an assertion or has a contract~~
~~condition (9.11.4), it is implementation-defined under which conditions all definitions of D shall be~~
~~translated using the same build level and violation continuation mode; and~~

In **[expr.const]/4**, remove a bullet:

~~a checked contract (9.11.4) whose predicate evaluates to false;~~

In **[dcl.attr.grammar]/1**, remove contract-attribute-specifier:

[[attribute-using-prefixopt attribute-list]]
~~contract-attribute-specifier~~
alignment-specifier

In **[dcl.attr.contract]**, remove the whole subclause.

In **[class.mem]/6**, remove a bullet:

~~contract condition (9.11.4), or~~

In **[class.virtual]/19**, remove the whole paragraph.

~~If an overriding function specifies contract conditions (9.11.4), it shall specify the same list of contract conditions...~~

In **[temp.expl.spec]/15**, remove the whole paragraph:

~~[Note: For an explicit specialization of a function template, the contract conditions (9.11.4) of the explicit specialization are independent of those of the primary template. — end note]~~

In **[except.terminate]/1**, remove three bullets:

~~when evaluation of the predicate of a contract (9.11.4) exits via an exception, or~~

~~when the violation handler invoked for a failed contract condition check (9.11.4) on a noexcept function exits via an exception, or~~

~~when the violation handler has completed after a failed contract check and the continuation mode is off, or~~

In **[tab:cpp.cond.ha]** remove the following entries from the table:

~~`assert 201806L`~~
~~`ensures 201806L`~~
~~`expects 201806L`~~

In **[macro.names]/2**, edit as follows:

to the attribute-tokens described in 9.11 ~~, or to the identifiers `expects` or `ensures`~~, except that the names likely and unlikely

In **[tab:headers.cpp]**, remove <contract> from the table.

In **[structure.specifications]/3**, edit as follows:

(3.4) — Expects: the conditions (sometimes termed preconditions) that the function assumes to hold whenever it is called. ~~[Example: An implementation might express such conditions via an attribute such as [[expects]] (9.11.4). However, some such conditions might not lend themselves to expression via code. —end example]~~

In **[support.general]/2**, edit as follows:

The following subclauses describe common type definitions used throughout the library, characteristics of the predefined types, functions supporting start and termination of a C ++ program, support for dynamic memory management, support for dynamic type identification, ~~support for contract violation handling,~~ support for exception processing...

In **[tab:support.summary]**, remove `<contract>` from the table.

In **[support.contract]**, remove the whole subclause..

In **[temp.dep.expr]** remove bullet (3.7):

~~— the identifier introduced in a postcondition (9.11.4) to represent the result of a templated function whose declared return type contains a placeholder type,~~