`when_all()` is just `just()`

Document Number: P4217R1
Date: 2026-06-23
Reply-to: Robert Leahy <rleahy@rleahy.ca>
Audience: LEWG

Abstract

This paper proposes giving `std::execution::when_all()` the same meaning as `std::execution::just()`.

Background

The standard currently specifies, by fiat, that `std::execution::when_all()` is ill-formed (§33.9.12.12 [exec.when.all]):

"The expressions `when_all(sndrs...)` and `when_all_with_variant(sndrs...)` are ill-formed if any of the following is `true`:

- *`sizeof...(sndrs)` is `0`, or*
- *[...]"*

If this restriction were not present the asynchronous operation which results from connecting the result of `std::execution::when_all()` and starting the operation state yielded thereby would hang due to the fact the implementation of `std::execution::start` therefor does not complete the operation eagerly when there are no child senders (ibid.).

Discussion

Behavior

`std::execution::when_all` "*adapt[s] multiple input senders into a sender that completes when all input senders have completed*" (ibid.). Given zero senders "all input senders" have always trivially completed (in the same way that `std::all_of` returns `true` for empty input) and therefore there's no reason to ban `std::execution::when_all()`, it is simply equivalent to `std::execution::just()`.

Banning `std::execution::when_all()` (i.e. the status quo) unnecessarily creates a special case when writing generic algorithms.

Coalescing to `just()`

This paper proposes having the `std::execution::when_all()` return `std::execution::just()` rather than *`make-sender`*`(when_all, {})`. This approach has the advantage that customizations of `std::execution::when_all` (including the implementation provided by the standard library itself) don't need to concern themselves with the `sizeof...(sndrs) == 0` case. The disadvantage is that destructuring such a sender yields a tag of type `std::execution::just_t` rather than `std::execution::when_all_t`.

During SG1 review in Brno 2026 SG1 requested that the author solicit feedback on this issue from Ben Deane and/or Michael Caisse, maintainers of Intel's "Bare Metal Senders and Receivers" [1]. Ben Deane provided the following:

"a) Yes `when_all()` is well-formed for us and completes immediately and synchronously on the value channel as you would expect.

"b) It is not implemented with `just()`. We have a system of debug signals for senders, so this is a (weak?) argument in favour of preserving types as written.

"Having said that, we do implement `when_all(s)` (the unary case) by coalescing to `s`."

SG1 proceeded to forward this paper with the coalescing to `std::execution::just()` with the understanding that the above context would be delivered to LEWG. The paper has been updated to include an alternate wording option which does not perform the coalescing.

Proposal

[exec.when.all]

The following change is to be made unconditionally:

[...]

The names `when_all` and `when_all_with_variant` denote customization point objects. Let `sndrs` be a pack of subexpressions and let `Sndrs` be a pack of the types `decltype((sndrs))...`. The expressions `when_all(sndrs...)` and `when_all_with_variant(sndrs...)` are ill-formed if ~~any of the following is `true`:~~

- ~~`sizeof...(sndrs)` is `0`, or~~
- `(sender<Sndrs> && ...)` is `false`.

The expression `when_all(sndrs...)` is expression-equivalent to.

[...]

Coalescing

The following change represents the coalesce-to-`std::execution::just()` approach (forwarded to LEWG by SG1):

[...]

The expression `when_all(sndrs...)` is expression-equivalent to:

- *`make-sender`*`(when_all, {}, sndrs...)` if `sizeof...(sndrs)` is not `0`, or
- `just()` otherwise.

[...]

Non-Coalescing

The following change does not coalesce to `std::execution::just()` (alternate approach to that which was forwarded to LEWG by SG1):

[...]

The member *`impls-for`*`<when_all_t>::start` is initialized with a callable object equivalent to the following lambda expression:

```
[]<class State, class Rcvr, class... Ops>(
    State& state, Rcvr& rcvr, Ops&... ops) noexcept -> void {
  if constexpr (sizeof...(Ops)) {
    state.on_stop.emplace(
      get_stop_token(get_env(rcvr)),
      on-stop-request{state.stop_src});
    (start(ops), ...);
  } else {
    set_value(std::move(rcvr));
  }
}
```

[...]

Implementation Experience

The coalescing option has been implemented against nVidia's reference implementation of `std::execution` [2].

Review History

Brno 2026

Paper was seen by SG1 in the first afternoon session 2026-06-10.

Concerns were raised about coalescing `when_all()` to `just()` (rather than providing an implementation of `when_all` which admits zero child senders).

The following poll was taken:

POLL: Forward P4217R0 to LEWG for C++29.

SF	F	N	A	SA
5	5	0	0	0

Attendance: Not recorded
# of Authors: 1
Author's Position: SF
Outcome: Unanimous consent

Revision History

R1

- Added review history since paper was seen by SG1
- Added discussion of coalescing to `std::execution::just()` (including context from Ben Deane as requested by SG1)
- Added alternate wording for an approach which does not coalesce to `std::execution::just()`
- Formatting fix

Acknowledgements

The author would like to thank:

- Jonathan Müller for his C++Now 2026 talk which made him aware of this defect
- Ben Deane for context from his implementation of `when_all`

References

[1] https://github.com/intel/cpp-baremetal-senders-and-receivers
[2] https://github.com/NVIDIA/stdexec/pull/2124