Document No: WG21 N4666
Date: 2017-06-05
Project: Programming Language C++
References:
Reply to: Barry Hedquist <beh@peren.com>
INCITS/PL22.16 IR

National Body Comments: ISO/IEC PDTS 22277, C++ Extensions for Coroutines

Attached is SC22 N5205, a complete set of National Body Comments submitted to JTC1 SC22 in response to the SC22 N5193, Ballot for ISO/IEC PDTS 22277, C++ Extensions for Coroutines.

These comments are to be addressed at the next WG21 meeting in Toronto, July 10 - 15, 2017.

Document numbers referenced in the ballot comments are WG21 documents unless otherwise stated.

# ISO/IEC JTC 1/SC 22 N 5205

2017-05-26

## ISO/IEC JTC 1/SC 22
## Programming languages, their environments and system software interfaces

**Document Type:** Summary of Voting

**Document Title:** Summary of Voting on PDTS 22277, Technical Specification -- C++ Extensions for Coroutines

**Source:** SC 22 Secretariat

**Document Status:** The PDTS ballot received 100% approval with comments from Canada, Switzerland and the United States.

The ballot results and the accompanying comments are forwarded to the SC 22/WG 21 July 2017 meeting in Toronto for review and resolution of the comments and preparation of an approved disposition of comments document, a revised text, and a recommendation for further processing.

The Project Editor is instructed to prepare the proposed disposition of comments document as soon as possible.

**Action ID:** INFO

**Due Date:**

**No. of Pages:**

Secretariat, ISO/IEC JTC 1/SC 22, American National Standards Institute, 25 West 43rd Street, New York, NY 10036; Telephone: 1 212 642 8934; Facsimile: 1 212 840 2298; Email: mmiller@ansi.org

# Result of voting

| Ballot Information | |
|---|---|
| **Ballot reference** | ISO/IEC PDTS 22277 |
| **Ballot type** | DTS |
| **Ballot title** | Technical Specification -- C++ Extensions for Coroutines |
| **Opening date** | 2017-03-28 |
| **Closing date** | 2017-05-23 |
| **Note** | |

| Member responses: | |
|---|---|
| **Votes cast (19)** | Austria (ASI)<br>Canada (SCC)<br>China (SAC)<br>Denmark (DS)<br>Finland (SFS)<br>France (AFNOR)<br>Germany (DIN)<br>Italy (UNI)<br>Japan (JISC)<br>Kazakhstan (KAZMEMST)<br>Korea, Republic of (KATS)<br>Netherlands (NEN)<br>Russian Federation (GOST R)<br>Slovenia (SIST)<br>Spain (UNE)<br>Switzerland (SNV)<br>Ukraine (DSTU)<br>United Kingdom (BSI)<br>United States (ANSI) |
| **Comments submitted (2)** | Egypt (EOS)<br>India (BIS) |
| **Votes not cast (1)** | Bulgaria (BDS) |

| Questions: | |
|---|---|
| **Q.1** | "Do you approve the draft for publication?" |

| Votes by members | Q.1 |
|---|---|
| **Austria (ASI)** | Abstention |
| **Canada (SCC)** | Approval with comments |
| **China (SAC)** | Approval |
| **Denmark (DS)** | Approval |

| | |
|---|---|
| **Finland (SFS)** | Abstention |
| **France (AFNOR)** | Approval |
| **Germany (DIN)** | Approval |
| **Italy (UNI)** | Abstention |
| **Japan (JISC)** | Approval |
| **Kazakhstan (KAZMEMST)** | Abstention |
| **Korea, Republic of (KATS)** | Approval |
| **Netherlands (NEN)** | Approval |
| **Russian Federation (GOST R)** | Approval |
| **Slovenia (SIST)** | Approval |
| **Spain (UNE)** | Approval |
| **Switzerland (SNV)** | Abstention |
| **Ukraine (DSTU)** | Approval |
| **United Kingdom (BSI)** | Approval |
| **United States (ANSI)** | Approval with comments |

| **Answers to Q.1: "Do you approve the draft for publication?"** | | |
|---|---|---|
| **12 x** | **Approval** | **China (SAC)**<br>**Denmark (DS)**<br>**France (AFNOR)**<br>**Germany (DIN)**<br>**Japan (JISC)**<br>**Korea, Republic of (KATS)**<br>**Netherlands (NEN)**<br>**Russian Federation (GOST R)**<br>**Slovenia (SIST)**<br>**Spain (UNE)**<br>**Ukraine (DSTU)**<br>**United Kingdom (BSI)** |
| **2 x** | **Approval with comments** | **Canada (SCC)**<br>**United States (ANSI)** |
| **0 x** | **Disapproval** | |
| **5 x** | **Abstention** | **Austria (ASI)**<br>**Finland (SFS)**<br>**Italy (UNI)**<br>**Kazakhstan (KAZMEMST)**<br>**Switzerland (SNV)** |

| **Comments from Voters** |
|---|

| Member: | Comment: | Date: |
|---|---|---|
| **Canada** (SCC) | ***Comment File*** | 2017-05-08 22:03:55 |
| CommentFiles/ISO_IEC PDTS 22277_SCC.doc | | |
| **Switzerland** (SNV) | ***Comment File*** | 2017-05-15 11:16:49 |
| CommentFiles/ISO_IEC PDTS 22277_SNV.doc | | |
| **United States** (ANSI) | ***Comment File*** | 2017-05-08 22:03:40 |
| CommentFiles/ISO_IEC PDTS 22277_ANSI.docx | | |

| **Comments from Commenters** | | |
|---|---|---|
| Member: | Comment: | Date: |
| **Egypt (EOS)** | ***Comment*** | 2017-04-06 12:15:13 |
| ok | | |
| **India (BIS)** | ***Comment*** | 2017-05-01 06:37:37 |
| Approved as presented | | |

## Template for comments and secretariat observations

| Date:2017-05-25 | Document: | Project: |
|---|---|---|

| MB/ NC[1] | Line number | Clause/ Subclause | Paragraph/ Figure/Table | Type of comment[2] | Comments | Proposed change | Observations of the secretariat |
|---|---|---|---|---|---|---|---|
| CH 001 | | | | te | This TS disallows stackful coroutines. This is too restrictive and stackful coroutines should be allowed as well. | Allow as suspension context functions that were called from a top-level coroutine. | |
| US 002 | na | 05.03.8 | 4 | Ed | in<br><br>"The await-expression has the same type and value category as the await-resume expression."<br>await-resume is marked up in bold, should be italics. | Change to italics. | |
| US 003 | na | 06.05.4 | 1 | Te | Update range based for statement after C++17 | The range-based for statement<br>for co_await$_{opt}$<br>( *for-range-declaration* : *for-range-initializer* )<br>statement<br>is equivalent to<br>{<br>auto &&__range = *for-range-initializer* ;<br>auto __begin =<br>co_await$_{opt}$ begin-expr ;<br>auto __end = end-expr ;<br>for ( ; __begin != __end; co_await$_{opt}$ ++__begin ) {<br>*for-range-declaration* = *__begin;<br>*statement*<br>}<br>} | |
| US 004 | na | 06.06.3 | all | Ge | There are many new cases of undefined behaviour introduced by the TS which are somewhat easily | No action for now. However, experience with TS implementation may allow reducing UB. This should | |

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

| MB/ NC[1] | Line number | Clause/ Subclause | Paragraph/ Figure/Table | Type of comment[2] | Comments | Proposed change | Observations of the secretariat |
|---|---|---|---|---|---|---|---|
| | | 6.6.3.1<br>8.4.4<br>8.11.2.5<br>18.10<br>18.11.2.5 | | | triggered by independent parts of the mechanisms, e.g., the result type of the coroutine interacting through the promise_type to allow flow of control to run off the end of a coroutine.<br>In general it would be good to minimize undefined behaviour. | form part of any review for integrating coroutines as part of a future standard. | |
| US 005 | na | 06.06.3.1 | 1 | Te | Simplify the grammar for<br>*coroutine-return-statement:*<br>*co_return expression_opt_;*<br>*co_return braced-init-list;* | coroutine-return-statement:<br>co_return *expr-or-braced-init-list* opt ; | |
| US 006 | na | 08.04.4 | 12 | Technical | Stateful allocators (pmr) do not work this way, there's no mechanism for allocator propagation to the captured state. | Strike section 12, or provide mechanism for holding allocator | |
| US 007 | na | 08.04.4 | 3 | Ge | Is unhandled_exception() a requirement for a promise_type? | a) Call std::terminate if not present<br>or<br>b) Add unhandled_exception() to the complete example of promise_type in 8.4.4 paragraph 11, the generator example. | |

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

## Template for comments and secretariat observations

| Date:2017-05-25 | Document: | Project: |
|---|---|---|

| MB/ NC[1] | Line number | Clause/ Subclause | Paragraph/ Figure/Table | Type of comment[2] | Comments | Proposed change | Observations of the secretariat |
|---|---|---|---|---|---|---|---|
| US 008 | 4 | 08.04.4, p15 | 11 | Ed | Note about possibly undefined behaviour | If a coroutine has a parameter passed by reference, resuming the coroutine<br>after the lifetime of the entity referred to by that parameter has ended is ~~likely to result in~~<br>undefined behavior<br>Strike "likely to result in" | |
| CA 009 | | 18.01 [support.general] | Table 30 | ed | The entry for subclause 18.11 appears before the entry for subclause 18.10. | Move the insertion of the entry for subclause 18.11 to appear after the entry for subclause 18.10. | |
| US 010 | na | 18.11.01.1 | 1 | Te | Is the template coroutines_traits intended to be a user-extension point? If so, spell out the contract for users to customize this trait. Otherwise, restrict user specialization with the wording for all type traits in the <type_traits> header. 18.11.1p2 suggests the former, while the latter is much simpler to specify for the initial TS. | Specify the exact behaviour of user-customization of coroutine_traits. | |
| US 011 | na | 18.11.02 | all | Ge | The specification of each operation is not explicitly clear whether it applied to the specialization of coroutine_handle<void>, or the primary coroutine_handle template. | Break this section into two, to clearly provide definitions for both versions of the template. | |
| US 012 | na | 18.11.02 | all | Ge | Coroutine handles have essentially raw pointer semantics. Should there be a library type as part of the TS that does destroy / set to nullptr ? | If a library type is needed, please add it. | |
| US 013 | na | 18.11.02 | all | Ge | Promise types are required to implement either return_value() or return_void(), but not both, and it is undefined behaviour for a coroutine to run off the end, where return_void would be called. | Consider implementing both either_return() and return_value() for promise types, and eliminate the undefined behaviour. | |

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

| MB/ NC[1] | Line number | Clause/ Subclause | Paragraph/ Figure/Table | Type of comment[2] | Comments | Proposed change | Observations of the secretariat |
|---|---|---|---|---|---|---|---|
| | | | | | Why not allow both? It could make types that implement the promise_type contract more reuseable. | | |
| US 014 | na | 18.11.02.5 | 6 | Ed | a concurrent resumption of a coroutine by multiple threads may result in a data race | Possibly means concurrent destruction here, in the destroy method. | |
| US 015 | na | 18.11.02.7 | all | Te | As coroutine_handle<void> is a literal type, should the comparison operators be constexpr? | Add constexpr to the declaration/definition of operator==, operator !=, operator<, operator<=, operator>=, and operator> for arguments of type coroutine_handle<>. | |
| US 016 | na | 18.11.03 | 1 | Te | The names suspend_never and suspend_always should be (inline) constexpr variables of type suspend_never_t and suspend_always_t respectively. | Change suspend_never and suspend_always as appropriate. | |
| US 017 | na | 2 [intro.refs ] | 1 | Ge | We are in the process of balloting the final text of the next C++ standard, provisionally ISO/IEC 14882:2017.<br><br>We should hold back publishing this TS long enough to rebase on the text of the new standard.<br><br>Other than updating this reference, the change is almost entirely updating section numbers and cross-references.<br><br>The normative changes would be | The following referenced document is indispensable for the application of this document. For dated references,<br>only the edition cited applies.<br>(1.1) — ISO/IEC 14882:2014**7**, *Programming Languages – C++*<br>ISO/IEC 14882:2014**7** is hereafter called the C++ Standard. ...<br><br>(Still to add a mapping of section numbers and stable-refs)<br>(File separate comments on the noted list of issues - such as range-for below) | |

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

| MB/ NC[1] | Line number | Clause/ Subclause | Paragraph/ Figure/Table | Type of comment[2] | Comments | Proposed change | Observations of the secretariat |
|---|---|---|---|---|---|---|---|
| | | | | | updating the range based 'for' loop syntax;<br><br>the text for a 'return' statement would need adjusting;<br><br>the wording on restrictions with respect to longjmp should be reviewed;<br><br>hash support for coroutine_handle should be updated with the “enabled” terminology. | | |
| CA 018 | | 2 [intro.refs] | | ed | The form required by ISO/IEC Directives, Part 2, 2016 subclause 15.5.1 is not followed. | Use the text provided by the Directives. | |
| US 019 | na | All | all | Ge | The TS presents only low level mechanisms to implement coroutines. For final release in a C++ standard, standard library implementations of generators, futures from coroutines, guard types for handles, etc. should also ship. | Please consider adding standard library implementations of generators, futures from Coroutines, guard types for handles and any others that may be needed when Coroutines are incorporated into the C++ Standard. | |
| US 020 | na | All | all | Ge | Coroutines are invokable types, can they be stored by a std::function? What about a std::function<void()> that discards the result on invocation? | Disallow storing coroutines in std::function objects that discard their result. | |

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

**Template for comments and secretariat observations**

| Date:2017-05-25 | Document: | Project: |
|---|---|---|

| MB/ NC[1] | Line number | Clause/ Subclause | Paragraph/ Figure/Table | Type of comment[2] | Comments | Proposed change | Observations of the secretariat |
|---|---|---|---|---|---|---|---|

D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 22277_ANSI.docx: Collation successful

D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 22277_SCC.doc: Collation successful

D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 22277_SNV.doc: Collation successful

Collation of files was successful. Number of collated files: 3

SELECTED (number of files): 3

PASSED TEST (number of files): 3

FAILED TEST (number of files): 0

CCT - Version 4.0/2015

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial