PD ISO/IEC TS 19841:2015

BSI Standards Publication

Technical Specification for C++ Extensions for Transactional Memory

bsi.

...making excellence a habit.™

National foreword

This Published Document is the UK implementation of ISO/IEC TS 19841:2015.

The UK participation in its preparation was entrusted to Technical Committee IST/5, Programming languages, their environments and system software interfaces.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Published by BSI Standards Limited 2015

ISBN 978 0 580 89249 3

ICS 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 30 September 2015.

Amendments/corrigenda issued since publication

Date	Text affected

PD ISO/IEC TS 19841:2015

TECHNICAL SPECIFICATION

ISO/IEC TS 19841

First edition
2015-10-01

Technical Specification for C++ Extensions for Transactional Memory

Spécification technique pour les extensions C++ de la mémoire transactionnelle

Reference number
ISO/IEC TS 19841:2015(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

1 General . . . 6
1.1 Scope . . . 6
1.2 Acknowledgements . . . 6
1.3 Normative references . . . 6
1.4 Implementation compliance . . . 6
1.5 Feature testing . . . 6
1.10 Multi-threaded executions and data races . . . 7
2 Lexical conventions . . . 9
2.11 Identifiers . . . 9
2.12 Keywords . . . 9
4 Standard conversions . . . 10
4.3 Function-to-pointer conversion . . . 10
4.14 Transaction-safety conversion . . . 10
5 Expressions . . . 11
5.1 Primary expressions . . . 11
5.1.2 Lambda expressions . . . 11
5.2 Postfix expressions . . . 11
5.2.2 Function call . . . 11
5.2.9 Static cast . . . 12
5.10 Equality operators . . . 12
5.16 Conditional operator . . . 12
6 Statements . . . 13
6.6 Jump statements . . . 13
6.9 Synchronized statement . . . 13
6.10 Atomic statement . . . 14
7 Declarations . . . 15
7.4 The asm declaration . . . 15
7.6 Attributes . . . 15
7.6.6 Attribute for optimization in synchronized blocks . . . 15
8 Declarators . . . 16
8.3 Meaning of declarators . . . 16
8.3.5 Functions . . . 16
8.4 Function definitions . . . 17
8.4.1 In general . . . 17
8.4.4 Transaction-safe function . . . 17
10 Derived classes . . . 19
10.3 Virtual functions . . . 19
13 Overloading . . . 20
13.1 Overloadable declaration . . . 20
13.3 Overload resolution . . . 20
13.3.3 Best viable function . . . 20
13.3.3.1 Implicit conversion sequences . . . 20
13.3.3.1.1 Standard conversion sequences . . . 20
13.4 Address of overloaded function . . . 20
14 Templates . . . 21
14.1 Template parameters . . . 21
14.7 Template instantiation and specialization . . . 21
14.7.3 Explicit specialization . . . 21
14.8 Function template specializations . . . 21
14.8.2 Template argument deduction . . . 21
14.8.2.1 Deducing template arguments from a function call . . . 21
15 Exception handling . . . 22
15.1 Throwing an exception . . . 22
15.2 Constructors and destructors . . . 22

15.3 Handling an exception . . . 22
15.4 Exception specifications . . . 23
17 Library introduction . . . 24
17.5 Method of description (Informative) . . . 24
17.5.1 Structure of each clause . . . 24
17.5.1.4 Detailed specifications . . . 24
17.6 Library-wide requirements . . . 24
17.6.3 Requirements on types and expressions . . . 24
17.6.3.5 Allocator requirements . . . 24
17.6.5 Conforming implementations . . . 2
17.6.5.16 Transaction safety . . .
18 Language support library . . . 25
18.5 Start and termination . . . 25
18.6 Dynamic memory management . . . 25
18.6.1 Storage allocation and deallocation . . . 25
18.6.2 Storage allocation errors . . . 25
18.6.2.1 Class bad_alloc . . . 25
18.6.2.2 Class bad_array_new_length . . . 25
18.7 Type identification . . . 25
18.7.2 Class bad_cast . . . 25
18.7.3 Class bad_typeid . . . 26
18.8 Exception handling . . . 26
18.8.1 Class exception . . . 26
18.8.2 Class bad_exception . . . 26
18.10 Other runtime support . . . 26
19 Diagnostics library . . . 27
19.2 Exception classes . . . 27
19.2.10 Class template tx_exception . . . 27
20 General utilities library . . . 28
20.2 Utility components . . . 28
20.2.4 forward/move helpers . . . 28
20.7 Memory . . . 28
20.7.3 Pointer traits . . . 28
20.7.3.2 Pointer traits member functions . . . 28
20.7.5 Align . . . 28
20.7.8 Allocator traits . . . 29
20.7.8.2 Allocator traits static member functions . . . 29
20.7.9 The default allocator . . . 29
20.7.9.1 allocator members . . . 29
20.7.11 Temporary buffers . . . 29
20.7.12 Specialized algorithms . . . 29
20.7.12.1 addressof . . . 29
20.7.13 C library . . . 29
20.8 Smart pointers . . . 30
20.8.1 Class template unique_ptr . . . 30
21 Strings library . . . 31
21.1 General . . . 31
21.4 Class template basic_string . . . 31
21.4.3 basic_string iterator support . . . 31
21.4.4 basic_string capacity . . . 31
21.4.5 basic_string element access . . . 31
23 Containers library . . . 32
23.2 Container requirements . . . 32
23.2.1 General container requirements . . . 32
23.2.3 Sequence containers . . . 32
23.2.5 Unordered associative containers . . . 32
23.3 Sequence containers . . . 33
23.3.2 Class template array . . . 33

23.3.2.1 Class template array overview . . . 33
23.3.3 Class template deque . . . 33
23.3.3.1 Class template deque overview . . . 33
23.3.4 Class template forward_list . . . 33
23.3.4.1 Class template forward_list overview . . . 33
23.3.4.6 forward_list operations . . . 33
23.3.5 Class template list . . . 33
23.3.5.1 Class template list overview . . . 33
23.3.5.5 list operations . . . 33
23.3.6 Class template vector . . . 3
23.3.6.1 Class template vector overview . . .
23.3.6.3 vector capacity . . . 34
23.3.6.4 vector data . . . 34
23.3.7 Class vector<bool> . . . 34
23.4 Associative containers . . . 34
23.4.4 Class template map . . . 34
23.4.4.1 Class template map overview . . . 34
23.4.5 Class template multimap . . . 34
23.4.5.1 Class template multimap overview . . . 34
23.4.6 Class template set . . . 34
23.4.6.1 Class template set overview . . . 34
23.4.7 Class template multiset . . . 34
23.4.7.1 Class template multiset overview . . . 34
23.5 Unordered associative containers . . . 35
23.5.4 Class template unordered_map . . . 35
23.5.4.1 Class template unordered_map overview . . . 35
23.5.5 Class template unordered_multimap overview . . . 35
23.5.5.1 Class template unordered_multimap overview . . . 35
23.5.6 Class template unordered_set . . . 35
23.5.6.1 Class template unordered_set overview . . . 35
23.5.7 Class template unordered_multiset . . . 35
23.5.7.1 Class template unordered_multiset overview . . . 35
23.6 Container adaptors . . . 35
23.6.1 In general . . . 35
24 Iterators library . . . 36
24.4 Iterator primitives . . . 36
24.4.4 Iterator operations . . . 36
24.5 Iterator adaptors . . . 36
24.5.1 Reverse iterators . . . 36
24.5.2 Insert iterators . . . 36
24.5.3 Move iterators . . . 36
24.7 range access . . . 36
25 Algorithms library . . . 37
25.1 General . . . 37
26 Numerics library . . . 38
26.7 Generalized numeric operations . . . 38
26.7.1 Header <numeric> synopsis . . . 38
26.8 C library . . . 38

 N4514

Technical Specification for C++ Extensions for Transactional Memory

1 General [intro]

1.1 Scope [general.scope]

1 This Technical Specification describes extensions to the C++ Programming Language (1.3) that enable the specification of Transactional Memory. These extensions include new syntactic forms and modifications to existing language and library.

2 The International Standard, ISO/IEC 14882, provides important context and specification for this Technical Specification. This document is written as a set of changes against that specification. Instructions to modify or add paragraphs are written as explicit instructions. Modifications made directly to existing text from the International Standard use **green** to represent added text and ~~strikethrough~~ to represent deleted text.

3 This Technical Specification is non-normative. Some of the functionality described by this Technical Specification may be considered for standardization in a future version of C++, but it is not currently part of any C++ standard. Some of the functionality in this Technical Specification may never be standardized, and other functionality may be standardized in a substantially changed form.

4 The goal of this Technical Specification is to build widespread existing practice for Transactional Memory. It gives advice on extensions to those vendors who wish to provide them.

1.2 Acknowledgements [general.ack]

1 This work is the result of collaboration of researchers in industry and academia, including the Transactional Memory Specification Drafting Group and the follow-on WG21 study group SG5. We wish to thank people who made valuable contributions within and outside these groups, including Hans Boehm, Justin Gottschlich, Victor Luchangco, Jens Maurer, Paul McKenney, Maged Michael, Mark Moir, Torvald Riegel, Michael Scott, Tatiana Shpeisman, Michael Spear, Michael Wong, and many others not named here who contributed to the discussion.

1.3 Normative references [general.references]

1 The following referenced document is indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

— ISO/IEC 14882:2014, *Programming Languages - C++*

2 ISO/IEC 14882:2014 is hereinafter called the *C++ Standard*. Beginning with section 1.10 below, all clause and section numbers, titles, and symbolic references in [brackets] refer to the corresponding elements of the C++ Standard. Sections 1.1 through 1.5 of this Technical Specification are introductory material and are unrelated to the similarly-numbered sections of the *C++ Standard*.

1.4 Implementation compliance [intro.compliance]

1 Conformance requirements for this specification are the same as those defined in section 1.4 [intro.compliance] of the *C++ Standard*. [*Note:* Conformance is defined in terms of the behavior of programs. — *end note*]

1.5 Feature testing [intro.features]

1 An implementation that provides support for this Technical Specification shall define the feature test macro in Table 1.

Table 1 -- Feature Test Macro

Name	Value	Header
`__cpp_transactional_memory`	201505	*predeclared*