

This is a preview - [click here to buy the full publication](#)

**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/IEC 2015

**COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2015, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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](mailto:copyright@iso.org)  
[www.iso.org](http://www.iso.org)

## Contents

<b>1</b>	<b>General</b>	<b>6</b>
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
<b>2</b>	<b>Lexical conventions</b>	<b>9</b>
2.11	Identifiers	9
2.12	Keywords	9
<b>4</b>	<b>Standard conversions</b>	<b>10</b>
4.3	Function-to-pointer conversion	10
4.14	Transaction-safety conversion	10
<b>5</b>	<b>Expressions</b>	<b>11</b>
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
<b>6</b>	<b>Statements</b>	<b>13</b>
6.6	Jump statements	13
6.9	Synchronized statement	13
6.10	Atomic statement	14
<b>7</b>	<b>Declarations</b>	<b>15</b>
7.4	The asm declaration	15
7.6	Attributes	15
7.6.6	Attribute for optimization in synchronized blocks	15
<b>8</b>	<b>Declarators</b>	<b>16</b>
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
<b>10</b>	<b>Derived classes</b>	<b>19</b>
10.3	Virtual functions	19
<b>13</b>	<b>Overloading</b>	<b>20</b>
13.1	Overloadable declarations	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
<b>14</b>	<b>Templates</b>	<b>21</b>
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
<b>15</b>	<b>Exception handling</b>	<b>22</b>
15.1	Throwing an exception	22
15.2	Constructors and destructors	22

15.3	Handling an exception . . . . .	22
15.4	Exception specifications . . . . .	23
<b>17</b>	<b>Library introduction . . . . .</b>	<b>24</b>
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 . . . . .	24
17.6.5.16	Transaction safety . . . . .	24
<b>18</b>	<b>Language support library . . . . .</b>	<b>25</b>
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
<b>19</b>	<b>Diagnostics library . . . . .</b>	<b>27</b>
19.2	Exception classes . . . . .	27
19.2.10	Class template tx_exception . . . . .	27
<b>20</b>	<b>General utilities library . . . . .</b>	<b>28</b>
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
<b>21</b>	<b>Strings library . . . . .</b>	<b>31</b>
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
<b>23</b>	<b>Containers library . . . . .</b>	<b>32</b>
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	33
23.3.6.1	Class template vector overview	33
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
<b>24</b>	<b>Iterators library</b>	<b>36</b>
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
<b>25</b>	<b>Algorithms library</b>	<b>37</b>
25.1	General	37
<b>26</b>	<b>Numerics library</b>	<b>38</b>
26.7	Generalized numeric operations	38
26.7.1	Header <numeric> synopsis	38
26.8	C library	38

# Technical Specification for C++ Extensions for Transactional Memory

## 1 General

[intro]

### 1.1 Scope

[general.scope]

- <sup>1</sup> 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.
- <sup>2</sup> 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.
- <sup>3</sup> 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.
- <sup>4</sup> 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]

- <sup>1</sup> 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]

- <sup>1</sup> 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++*
- <sup>2</sup> 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*.