Document No:	WG21 N4396
Date:	2015-03-16
References:	ISO/IEC PDTS 19568, SC22/5019
Reply To:	Barry Hedquist <beh@peren.com> INCITS/PL22.16 IR

National Body Comments

ISO/IEC PDTS 19841

C++ Extensions for Transactional Memory

This document contains the Result of Voting and National Body Comments for ISO/IEC PDTS 19841, C++ Extensions for Transactional Memory.

Document numbers referenced herein SC22/WG21 documents unless otherwise stated.

Template for comments and secretariat observations

Date:2015-03-12	Document: SC22/N 5019 WG21/N4396	Project: PDTS 19841

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
JP 1				ge	We are concerned that there could be a performance degradation even in an environment lacking of transactional memory feature by adopting this technical specification, e.g. making some standard libraries like <math.h> transactional safe. It is the reason for our disapproval. If we can reasonably confirm that there's no degradation, we will change our position to approval.	Please make us sure there's no degradation.	
US 1				te	Memory ordering requirements of transactions are problematically strict. Even empty or purely local transactions have observable synchronization effects and can usually not be removed by an optimizing compiler. This introduces a performance penalty when transactional library code is reused in a clearly thread-local context.	Consider weakening ordering requirements to allow such optimizations.	
CA 1	N/A	N/A	N/A	ge	Request to add a Feature Test Macro __cpp_transactional_memory based on http://isocpp.org/std/standing-documents/sd-6-sg10-feature-test-recommendations	The value of the macro will be the year and month of the release of the TS. It does not need any experimental or TS tag.	

CA 3	N/A	4.3 [conv.func]	Para 1	ge	Make helper functions in 20.2 transaction-safe. Here is an example where std::move is not transaction-safe
					template <class T>
					void safe_swap(T &a, T &b) transaction_safe
					{
					atomic_commit
					{
					using std::move;
					T temp = move(a); // Note that std::move is not transaction-safe according to draft, but it should be
					a = move(b);
					b = move(temp);
						Add std::move and other utilities in 20.2 to be transaction_safe.	

[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

					}
					}
					
					template <class T>
					void apply(T &a, T &b, void f(T&,&T))
					{
					f(a,b); // Ok
					assert(f == safe_swap<int>); // result unspecified according to 5.10, paragraph 2, right?
					}
					
					int main()
					{
					int x = 2, y = 3;
					apply(x, y, safe_swap<int>); // Ok even though transaction_safe is lost
					}		

CA 2	N/A	5.2.2 [expr.call]	Para 1	te	This addition states: A call to a virtual function that is evaluated within a synchronized (6.9 [stmt.sync]) or atomic block (6.10 [stmt.tx]) results in undefined behavior if the virtual function is declared transaction_safe_noinherit and the final overrider is not declared transaction_safe. It is Undefined Behavior if you call into a virtual function declared as tx_safe_noinherit but it is not tx_safe in the final overrider. This ensures that the dynamic call is safe, no matter what the dynamic object is since tx_safe_noinherit gives no such guarantee. Our concern is this is excessive for a synchronized block because these can call	Please fix for synchronized block so that it is not part of this requirement. Suggested wording: A call to a virtual function that is evaluated within ~~a synchronized (6.9 [stmt.sync]) or~~ an atomic block (6.10 [stmt.tx]) results in undefined behavior if the virtual function is declared transaction_safe_noinherit and the final overrider is not declared transaction_safe.	

[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
					tx_unsafe functions.		
FI 1		8	4	te	It seems that the applicability of transaction_safe_noinherit is likely going to be wider in the future than just in virtual functions. If that wider applicability appears, new keywords need to be added. Generalizing the name transaction_safe_noinherit would possibly avoid that problem.	Rename transaction_safe_noinherit to transaction_safe_dynamic. Transaction safety of calls to such functions is ultimately a runtime property, hence _dynamic seems like a suitable suffix.	
JP 2		8.4.4	1	te	A function-local static variable initialization should be transactional-unsafe. The initialization in an atomic execution needs to be synchronized with non-atomic executions.	Add "a function-local static variable initialization" in the list of conditions for a transactional-unsafe statement .	

CA 4	n/a	8.4.4
		[dcl.fct.def.tx]
			After Para 1, bullet 5	ge	In the first sequence of dash bullets (--) indicating transaction-unsafe expressions, the fifth one states «an implicit call of a non-virtual function that is not transaction_safe». I wonder why the «implicit» call is being explicitly (sorry for the pun! :)) specified, as it seems to me that an explicit call to a non-virtual function would yield the same consequences. Unless I'm missing out on something, an implicit call could be something like:
					
					struct B
					{
					   int f(); // not transaction_safe, not virtual
					   virtual ~B() = default;
					};
					
					struct D : B
					{
					   int g()
					   {
						This seems a possible confusion for other user, please clarify.	

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

					return f() + // implicit call?
					this->f() + // explicit call?
					B::f(); // explicit call?
					}
					};		

[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

Result of voting

Ballot Information	
Ballot reference	ISO/IEC PDTS 19841
Ballot type	CD
Ballot title	C++ Extensions for Transactional Memory
Opening date	2014-12-10
Closing date	2015-03-10
Note	Please submit your vote by the due date indicated.

Member responses:	
Votes cast (18)	Austria (ASI) Canada (SCC) China (SAC) Denmark (DS) Finland (SFS) Germany (DIN) Italy (UNI) Japan (JISC) Kazakhstan (KAZMEMST) Korea, Republic of (KATS) Netherlands (NEN) Portugal (IPQ) Russian Federation (GOST R) Spain (AENOR) Switzerland (SNV) Ukraine (DTR) United Kingdom (BSI) United States (ANSI)
Comments submitted (0)	
Votes not cast (0)	

Questions:	
Q.1	"Do you agree with approval of the CD text?"
Q.2	"If you approve the CD text with comments, would you please indicate which type ? (General, Technical or Editorial)"
Q.3	"If you disappove the draft, would you please indicate if you accept to change your vote to Approval if the reasons and appropriate changes will be accepted?"

Votes by members	Q.1	Q.2	Q.3
Austria (ASI)	Abstention	Ignore	Ignore

Canada (SCC)	Approval with comments	All	Ignore
China (SAC)	Approval as presented	Ignore	Ignore
Denmark (DS)	Abstention	Ignore	Ignore
Finland (SFS)	Approval with comments	Technical	Ignore
Germany (DIN)	Approval as presented	Ignore	Ignore
Italy (UNI)	Approval as presented	Ignore	Ignore
Japan (JISC)	Disapproval of the draft	Ignore	Yes
Kazakhstan (KAZMEMST)	Abstention	Ignore	Ignore
Korea, Republic of (KATS)	Approval as presented	General	Ignore
Netherlands (NEN)	Approval as presented	Ignore	Ignore
Portugal (IPQ)	Abstention	Ignore	Ignore
Russian Federation (GOST R)	Approval as presented	Ignore	Ignore
Spain (AENOR)	Approval as presented	Ignore	Ignore
Switzerland (SNV)	Approval as presented	Ignore	Ignore
Ukraine (DTR)	Approval as presented	Ignore	Ignore
United Kingdom (BSI)	Approval as presented	Ignore	Ignore
United States (ANSI)	Approval with comments	Technical	Ignore

Answers to Q.1: "Do you agree with approval of the CD text?"		
10 x	**Approval as presented**	**China (SAC) Germany (DIN) Italy (UNI) Korea, Republic of (KATS) Netherlands (NEN) Russian Federation (GOST R) Spain (AENOR) Switzerland (SNV) Ukraine (DTR) United Kingdom (BSI)**
3 x	**Approval with comments**	**Canada (SCC) Finland (SFS) United States (ANSI)**

1 x	Disapproval of the draft	Japan (JISC)
4 x	Abstention	Austria (ASI) Denmark (DS) Kazakhstan (KAZMEMST) Portugal (IPQ)

Answers to Q.2: "If you approve the CD text with comments, would you please indicate which type ? (General, Technical or Editorial)"		
1 x	General	Korea, Republic of (KATS)
2 x	Technical	Finland (SFS) United States (ANSI)
0 x	Editorial	
1 x	All	Canada (SCC)
14 x	Ignore	Austria (ASI) China (SAC) Denmark (DS) Germany (DIN) Italy (UNI) Japan (JISC) Kazakhstan (KAZMEMST) Netherlands (NEN) Portugal (IPQ) Russian Federation (GOST R) Spain (AENOR) Switzerland (SNV) Ukraine (DTR) United Kingdom (BSI)

Answers to Q.3: "If you disappove the draft, would you please indicate if you accept to change your vote to Approval if the reasons and appropriate changes will be accepted?"		
1 x	Yes	Japan (JISC)
0 x	No	
17 x	Ignore	Austria (ASI) Canada (SCC) China (SAC) Denmark (DS) Finland (SFS) Germany (DIN) Italy (UNI) Kazakhstan (KAZMEMST) Korea, Republic of (KATS) Netherlands (NEN) Portugal (IPQ) Russian Federation (GOST R) Spain (AENOR) Switzerland (SNV) Ukraine (DTR) United Kingdom (BSI)

United States (ANSI)

Comments from Voters		
Member:	Comment:	Date:
Canada (SCC)	***Comment File***	2015-02-11 19:38:20
CommentFiles/ISO_IEC PDTS 19841_SCC.doc		
Finland (SFS)	***Comment File***	2015-03-11 08:11:01
CommentFiles/ISO_IEC PDTS 19841_SFS.doc		
Japan (JISC)	***Comment File***	2015-03-10 03:05:27
CommentFiles/ISO_IEC PDTS 19841_JISC.doc		
United States (ANSI)	***Comment File***	2015-02-02 18:00:31
CommentFiles/ISO_IEC PDTS 19841_ANSI.doc		

Comments from Commenters		
Member:	Comment:	Date: