Document No: WG21 N4697
Date: 2017-10-07
Project: Programming Language C++
References: SC 22 N5250
Reply to: Barry Hedquist
INCITS/PL22.16 IR
beh@peren.com

SUMMARY and National Body Comments ISO/IEC PDTS 21544, C++ Extensions for Modules

Attached is the Summary of Voting and National Body Comments for ISO/IEC PDTS 21544, C++ Extensions for Modules.

The ballot results and the accompanying comments are forwarded to the SC 22/WG 21 November 2017 meeting in Albuquerque, NM, 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.

A key was added at the end of the ballot comments to indicate the designations used for those National Bodies submitting comments:

CA - Canada (SCC)

CH - Switzerland (SNV)

GB - Great Britain (BSI)

JP - Japan (JISC)

US - United States (ANSI)

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

ISO/IEC JTC 1/SC 22 N 5250

2017-10-06

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 21544, Technical Specification -- C++ Extensions for Modules
Source:	SC 22 Secretariat
Document Status:	The PDTS ballot was approved with one negative vote from Canada. The ballot results and the accompanying comments are forwarded to the SC 22/WG 21 November 2017 meeting in Albuquerque, NM, 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:	

Result of voting

Ballot Information	
Ballot reference	ISO/IEC PDTS 21544 - JTC001-SC22-N5233
Ballot type	DTS
Ballot title	Technical Specification -- C++ Extensions for Modules
Opening date	2017-08-08
Closing date	2017-10-03
Note	

Member responses:	
Votes cast (20)	Austria (ASI) Bulgaria (BDS) Canada (SCC) China (SAC) Denmark (DS) Finland (SFS) France (AFNOR) Germany (DIN) Italy (UNI) Japan (JISC) Kazakhstan (KAZMEMST) Korea, Republic of (KATS) Netherlands (NEN) Russian Federation (GOST R) Slovenia (SIST) Spain (UNE) Switzerland (SNV) Ukraine (DSTU) United Kingdom (BSI) United States (ANSI)
Comments submitted (0)	
Votes not cast (0)	

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

Votes by members	Q.1
Austria (ASI)	Abstention
Bulgaria (BDS)	Abstention
Canada (SCC)	Disapproval
China (SAC)	Approval

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

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

Comments from Voters		
Member:	Comment:	Date:
Canada (SCC)	***Comment File***	2017-09-21 19:43:07
CommentFiles/ISO_IEC PDTS 21544 - JTC001-SC22-N5233_SCC.doc		
Japan (JISC)	***Comment File***	2017-10-03 09:58:20
CommentFiles/ISO_IEC PDTS 21544 - JTC001-SC22-N5233_JISC.doc		
Switzerland (SNV)	***Comment File***	2017-09-25 10:38:58
CommentFiles/ISO_IEC PDTS 21544 - JTC001-SC22-N5233_SNV.doc		
United Kingdom (BSI)	***Comment File***	2017-09-19 17:04:00
CommentFiles/ISO_IEC PDTS 21544 - JTC001-SC22-N5233_BSI.doc		
United States (ANSI)	***Comment File***	2017-09-26 17:24:55
CommentFiles/ISO_IEC PDTS 21544 - JTC001-SC22-N5233_ANSI.docx		

Comments from Commenters		
Member:	Comment:	Date:

Template for comments and secretariat observations

Date:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
US 001				ge	We have concerns regarding the ability of tools (e.g., SWIG, static analyzers), to consume source code that contains module import declarations. We feel that a requirement must be added to ensure that it be possible to programmatically rewrite a module import declaration in terms of textual inclusion such that the included text (however obtained) matches the semantic behaviour of the module import declaration it replaces. This concern is motivated by observations that module artifacts produced by compilers are being (internally) distributed within real world build environments in lieu of source code. In such scenarios, tools are unable to construct their own module artifacts in order to satisfy module import declarations. We are hopeful that compiler implementers will be willing and able to provide tools that, given a module artifact, will generate source code suitable for use as a textual inclusion substitution for a module import declaration, or suitable for constructing a module artifact appropriate for the tool in question. The ability to do so depends on the requirement indicated above. We will follow up with a paper detailing this concern.	Add a requirement effectively stating that it must be possible to mimic the effects of any module import declaration with textual inclusion such that name lookup and overload resolution produce the same results.	
US 002				Ge	It is essential that the module design supports users deploying a phased adoption, retaining a non-module (#include) interface to their existing code along-side a parallel module-interface for newer clients. Remote clients need to be able to indirectly import the contents of such a module through both interfaces in the same translation unit.	As the #included interface will live in the global module, we need a means for an interface module to adopt and export a restricted subset of the global module. We will provide a more detailed paper with potential solutions before the Albuquerque meeting. It was previously suggested that simply 'using' the global module names would suffice, but that does not work with the TS as specified.	
US 003				Ed	There is no Annex A collecting all the grammar changes.	Add Annex A collecting all the grammar changes, corresponding to Annex A in the C++ standard.	

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:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
US 004				Ed	There is no compatibility annex.	Add Annex C for compatibility with C++17, at a minimum noting that new keywords remove previously valid identifiers from the users.	
US 005		01	paragraph 1	ed	It is customary to refer to Clauses with text of the form “Clause 2”.	Use “Clause 2” instead of “2”.	
CA 006		01	paragraph 2	ge	The normative interpretation of the document is established by the subject paragraph to require accurate perception of a specific colour. This barrier to accessibility was not present in documents such as ISO/IEC TS 19217:2015(E).	Follow the recommendations in Clause 4 of ISO/IEC Guide 71:2014(E). The specific barrier identified is listed as a design consideration under ISO/IEC Guide 71:2014(E) subclause 7.2.2.3. A possible mitigation is to include text markers for delimiting added text and deleted text.	
CA 007		01	paragraph 2	te	The editing instructions in the document do not apply to ISO/IEC 14882. As of this writing, the corresponding dated reference would be to ISO/IEC 14882:2014. The document, as presented, is not usable.	Refer to a suitable base document in an appropriate manner.	
US 008		02	paragraph 1	ed	The title given for the document in the subject paragraph does not match that of the referenced document (JTC 1/SC 22/WG 21 N 4660). Additionally, N4660 is not a unique document identifier. ISO/IEC DIS 14882:2017 is preferable.	Reference ISO/IEC DIS 14882:2017 appropriately.	
US 009		02	paragraph 1	ed	It is not customary to use the capitalization in “Clauses” as opposed to that of “clauses” when referring to clauses in general.	Replace “Clauses” with “clauses” in each instance within the subject paragraph.	

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:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
CA 010		02	paragraph 1	ed	The form required by ISO/IEC Directives, Part 2, 2016 subclause 15.5.1 is not followed.	Use the introductory text provided by the Directives.	
CA 011		03		ed	The introductory text from the ISO/IEC Directives, Part 2, 2016 subclause 16.5.2 is not present.	Use the introductory text provided by the Directives.	
US 012		04.01	paragraph 1	te	The wording does not clearly establish that conformance with the TS is to be interpreted as conformance with the document that results from applying the editing instructions to the base document as opposed to conformance with the vanilla base document.	Replace “C++ Standard” with “C++ Standard as modified by the editing instructions contained in this document”.	
CA 013		04.01	paragraph 1	te	The wording does not clearly establish that conformance with the TS is to be interpreted as conformance with the document that results from applying the editing instructions to the base document as opposed to conformance with the vanilla base document.	Replace “C++ Standard” with “C++ Standard as modified by the editing instructions contained in this document”.	
CA 014		05.11		ed	Presumably, the note in paragraph 1 of the subject subclause in the base document should be updated to no longer claim that the `export` keyword is unused.	Add an editing instruction to adjust the note appropriately.	
CA 015		05.11	paragraph 1	te	Table 3 does not exist within subclause 5.11 in WG 21 document N 4660.	Replace “Table 3” with “Table 5”.	
GB 016		06		Te	Modules should not be entities. Various wording changes throughout the TS make a module an entity, with a point of definition. This appears to achieve nothing and should be struck.	Revert the changes to 6/3, 6.1, 6.3.2. Remove the last sentence of 10.7/1 and the exclusion in 10.7/4: "A namespace-scope declaration D of an entity (other than a module)"	
GB		06.01		Ed	Change to 6.1 does not follow surrounding	Convert text to bulleted list. Provide the text prior to the bulleted list as context. Merge the new 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

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
017					formatting C++17 uses a bulleted list here. The proposed addition does not make sense, and should in any case include context showing how and where to add the specified text.	text into the existing example in p2.	
US 018		06.01	2	te	Clause 10 does not define a module-declaration or proclaimed-ownership-declaration as being a declaration (although the latter contains one) since they are the other possibilities for top-level-declaration.	Don't mention them here either.	
US 019		06.01 [basic.def]	2	Ed	This list is a bullet list in the latest draft of the standard, so the comma-separated list should be integrated into the bullet list.	Rewrite comma list as appending bullets to the current bullet list, using 'it is ' phrasing.	
US 020		06.02		ed	The first editing instruction under the 6.2 heading applies instead to 6.5 (as it says it does).	Move that text to section 6.5 in the TS.	
US 021		06.02		te	In the "seventh bullet" to be added: If all possible definitions of `D` appear in the purview of the same module, then this bullet is only reached if there is more than one definition of `D` in the owning module. The statement that there can be at most one definition of `D` in the owning module seems paradoxical. Even a friendly reading of this wording leaves questions over whether inline functions defined in the purview of a module in one module unit can be odr-used in other translation units, and similarly whether implicit instantiation may occur merely by importing or through the use of module linkage. As well, there seems to be deficiency in where class types may be used in a way that requires the class type to be complete.	Modify to instead add the new content to immediately before the sentence involving the list in the base document. Respectively replace the first and second instances of "`D`" with "such an entity" and "the entity". Replace "can" with "shall". Modify [dcl.inline] to adjust the requirement that an inline function or variable shall be defined in each translation unit in which it is odr-used. In particular, a definition in any module unit should suffice for an inline function with module linkage. In the case of an exported inline function, the aggregate result would be that there can only be one translation unit that exports the function.	

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
						It would make sense to For example, require that the definition be in the module interface unit if the inline function is odr-used in a translation unit other than the one where it is defined. Modify [basic.def.odr] to adjust the similar (but restricted to odr-use outside of a discarded statement) requirement. Similarly, modify [temp] to adjust the requirement that various forms of templated entities be defined in every translation unit in which it they are implicitly instantiated. As well, modify [basic.def.odr] to adjust the requirement that a definition of a class is required in a translation unit if the class is used such that the class type needs to be complete.	
GB 022		06.02		Ed	Edit to 6.5/3 needs moving The change "Modify bullet (3.2) of paragraph 6.5/3 as follows" appears in the section 6.2 One-definition rule [basic.def.odr] rather than in 6.5 Program and linkage [basic.link]	Move the change to 6.5/3 from section 6.2 into section 6.5	
JP 023		06.02		ed	A modification for the subclause 6.5 in the original document is described in the subclause 6.2.	Move to the subclause 6.5.	
CA 024		06.02		te	In the “seventh bullet” to be added: If all possible definitions of D appear in the purview of the same module, then this bullet is only reached if there is more than one definition of D in the owning module. The statement that there can be at most one definition of D in the owning module seems paradoxical. Even a friendly reading of this wording leaves questions over whether inline functions defined in the purview of a module in one module unit can be odr-used in other translation units, and similarly whether implicit instantiation may occur merely by importing or through the use	Modify to instead add the new content to immediately before the sentence involving the list in the base document. Respectively replace the first and second instances of “D” with “such an entity” and “the entity”. Replace “can” with “shall”. Modify [dcl.inline] to adjust the requirement that an inline function or variable shall be defined in each translation unit in which it is odr-used. For example, require that the definition be in the module interface unit if the inline function is odr-used in a translation unit other than the one where it is defined. Modify [basic.def.odr] to adjust the similar (but restricted to	

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
					of module linkage. As well, there seems to be deficiency in where class types may be used in a way that requires the class type to be complete.	odr-use outside of a discarded statement) requirement. Similarly, modify [temp] to adjust the requirement that various forms of templated entities be defined in every translation unit in which it they are implicitly instantiated. As well, modify [basic.def.odr] to adjust the requirement that a definition of a class is required in a translation unit if the class is used such that the class type needs to be complete.	
CA 025		06.02		ed	The editing instruction for a paragraph under subclause 6.5 in the base document appears out-of-place in subclause 6.2.	Move the editing instruction to subclause 6.5.	
US 026		06.02	6	te	This rule applies even to the global module, prohibiting all multiply-defined entities.	Write "purview of a named module" instead of "purview of a module".	
US 027		06.02	6	te	This new bullet is under the heading "Given such an entity named D defined in more than one translation unit, then", but prohibits multiple definitions of D.	Rephrase as a prohibition (for multiply-defined D) on appearing in a module at all.	
US 028		06.02	6	ge	Some users have described an important path to module adoption involving grouping existing components into modules without prejudicing their use via header files. Since exported entities have the same external linkage they have always had, the compatibility seems doable.	Change the rule to apply only when D's first declaration is in a named module. Alternatively, deliberately support the "export using" trick by allowing using-declarations in different modules.	
GB 029		06.02	6	Ed	New bullet in 6.2/6 does not fit enclosing context The context of 6.2's bullets is "Given such an entity named D defined in more than one translation unit, then". It does not make sense to follow this with a bullet that ends with "there can be at most one	Replace the bullet with: "there shall not be a definition of D within the purview of a module (10.7)" (Possibly add a note about module declaration if	

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

Date:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					definition of D". Also, it does not make sense to constrain declarations here. And that constraint is unnecessary since it is not possible to redeclare such an entity anywhere other than in the module interface or a proclaimed-ownership-declaration due to the linkage rules.	this seems unobvious)	
US 030		06.03.6		te	In paragraph 6.3.6/1 of the base document as modified by the PDTS: There appears to be no requirement that an exported name be declared in the module interface unit of its owning module. The wording in the subject paragraph appears to rely on such a requirement.	Require in [dcl.module] or a subclause thereof that an exported name be declared in the module interface unit. In the alternative, extend the potential scope of exported name *X0* of a member of a namespace *N0* (regardless of whether the name is declared in the module interface unit) as appropriate.	
US 031		06.03.6		te	In paragraph 6.3.6/1 of the base document as modified by the PDTS: There appears to be neither a requirement that a namespace is uniquely owned by a particular module, nor a requirement that a namespace be not in scope prior to importing a module that exports it. The wording in the subject paragraph appears to claim that the potential scope extends “backwards” from an import declaration. It is also unclear how the positioning of an import declaration interacts with the notion of “before” in unqualified name lookup.	Fully specify the effect of the positioning of an import declaration in [basic.scope] and [basic.lookup], or subclauses of the foregoing.	
CA 032		06.03.6		te	In paragraph 6.3.6/1 of the base document as modified by the PDTS: There appears to be no requirement that an exported name be declared in the module interface unit of its owning module. The wording in the subject paragraph appears to rely on such a requirement.	Require in [dcl.module] or a subclause thereof that an exported name be declared in the module interface unit. In the alternative, extend the potential scope of exported name *X0* of a member of a namespace *N0* (regardless of whether the name is declared in the module interface unit) as appropriate.	
CA 033		06.03.6		te	In paragraph 6.3.6/1 of the base document as modified by the PDTS: There appears to be neither a requirement that a namespace is uniquely owned by a particular module, nor a requirement that a namespace be	Fully specify the effect of the positioning of an import declaration in [basic.scope] and [basic.lookup], or subclauses of the foregoing.	

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
					not in scope prior to importing a module that exports it. The wording in the subject paragraph appears to claim that the potential scope extends “backwards” from an import declaration. It is also unclear how the positioning of an import declaration interacts with the notion of “before” in unqualified name lookup.		
US 034		06.03.6	1	te	The rule's use of namespace-definition (a grammar production) prevents it from applying in its own example.	Write "If a name X is declared in a namespace N in the module interface unit of a module M, the potential scope of X includes the namespace N in every module unit of M and, if the name X is exported, in every translation unit that imports M.".	
US 035		06.03.6	1	te	The potential scope of such an X is extended backwards in the interface unit.	Add "implementation" to "every module unit of M".	
US 036		06.03.6 [basic,scope. namespace]	1	Te	Example to illustrate exporting namespace members does not actually use namespace.	Add namespaces to the example, rather than exporting from the global namespace.	
US 037		06.03.6, 10.7.1	1, 1	te	Exported names should not be visible before a module is imported or declared (in an implementation unit; these two paragraphs disagree on this point).	Specify that the potential scope begins after such an import/declaration. Rely on that scope rather than on the names being "visible" (which is the stuff of [basic.scope.hiding]).	
CA 038		06.04.2		ed	With regards to the example being added to the second paragraph of the subclause in the base document, the line indicated as being ill-formed does not provide sufficient justification. In particular, the declaration of `g_impl` in namespace `Q`, an associated namespace of `Q::X`, is found in the template definition context of `g1`; the `g_impl` so declared is a candidate function according to WG	Change the example to reflect either additional reasoning for its claim of ill-formedness or remove said claim. Move the example to subclause 17.6.4 [temp.dep.res] or a subclause thereof.	

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
					21 N 4660 subclause 17.6.4.2 [temp.dep.candidate].		
GB 039		06.04.2	2	Ed	Inconsistent header name in example 6.4.2/2 The example in paragraph 6.4.2/2 has a comment "Header file X.h" but the code in the interface unit for M1 contains #include "H.h"	Change example to #include "X.h"	
US 040		06.04.2	4	ed	"module M other than the global module" is (now) unnecessarily circuitous.	Use "named module M".	
US 041		06.04.2	4	ge	It is surprising that ADL can see non-exported functions/templates, although there is some precedent in the form of invisible friend functions. Much more surprising is that it can see those whose names have internal linkage or none at all.	Restrict the visibility, or add an example justifying such insight on the part of ADL.	
US 042		06.05		te	A proclaimed-ownership-declaration can contain any non-defining declaration (e.g., a module-import-declaration, a static_assert-declaration, or an export-declaration).	Add semantic or (preferably) grammar restrictions to prevent nonsense.	
US 043		06.05		te	A proclaimed-ownership-declaration cannot refer to a member of any (non-global) namespace: it cannot appear in a namespace and cannot use a qualified-id because it would have to have already been declared (which is precluded by the new 6.2/6.7).	Allow a proclaimed-ownership-declaration to appear in a namespace.	
US 044		06.05		ge	A proclaimed-ownership-declaration requires the sort of NDR agreement-at-a-distance that modules are supposed to preclude.	Remove them until a concrete use case (e.g., an insurmountable performance problem) is known.	
US		06.05		te	It is unclear if entities are intended to have both	If the status quo of the normative text is intended,	

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**
045					module linkage and external linkage in some cases as is the status quo of the PDTS given the provisions of [basic.link]/4 in WG 21 N 4660. This extends to cases where it appears that an entity declared in the purview of a module may be found to have module linkage in one translation unit and not to have such linkage in another translation unit. Of particular interest is the interaction with [basic.link]/9 of WG 21 N 4660 with regards to whether declarations of names with module linkage are intended to declare different entities in different modules even if they would be required to declare the same entity if external linkage was involved.	add notes and examples to support the interpretation. If the status quo is not intended, modify the normative text to implement the intent.	
US 046		06.05		te	In the new paragraph to be added before paragraph 6.5/8 of the base document: Presumably, an entity that is exported should not be given module linkage. If it is possible to declare an exported entity with a non-exported declaration, then the wording results in module linkage in too many cases.	Replace “that is introduced by a non-exported declaration” with “that is not exported”.	
CA 047		06.05		ed	The other bullets of the list in paragraph 2 of [basic.link] in WG 21 N 4660 all describe the ability of names (plural) declared in other scopes to denote the same entity as the name being said to have linkage. The text of the new bullet to be added is not consistent with that aspect of the existing bullets.	In the bullet to be added, change “name” in “can be referred to by name from other scopes” to “names”.	
CA 048		06.05		te	It is unclear if entities are intended to have both module linkage and external linkage in some cases as is the status quo of the PDTS given the provisions of [basic.link]/4 in WG 21 N 4660. This extends to cases where it appears that an entity declared in the purview of a module may be found to have module linkage in one translation unit and not to have such linkage in another translation unit. Of particular interest is the interaction with [basic.link]/9 of WG 21 N 4660 with regards to whether declarations of names with module	If the status quo of the normative text is intended, add notes and examples to support the interpretation. If the status quo is not intended, modify the normative text to implement the intent.	

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
					linkage are intended to declare different entities in different modules even if they would be required to declare the same entity if external linkage was involved.		

CA 049		06.05		te	If it is intended that external linkage does not apply to names that are not exported, then the treatment of the language linkage of names should be reviewed.
					In the code below, partial specialization matching for `Q` depends on the language linkage of the name of `B::foo(int)`; in turn, that language linkage depends on whether the name has external linkage (N4660 subclause 10.5 [dcl.link]/4).
					Which function name is exported depends on the result of the partial specialization matching.
					`export module M;`
					`namespace A {`
					`  extern "C" void foo(int);`
					`}`
					`namespace B {`
					`  extern "C" void foo(int);`
					`  void foo(float);`
					`}`
					`extern "C" { typedef void (&ty)(int); }`
					
					`template <ty, ty>`
					`struct Q { using ty = int; };`
					
					`template <ty F> struct Q<F, F> {`
					`  using ty = float;`
					`};`
					
					`namespace A { export void foo(int); }`
					`namespace B {`
					`  export void foo(Q<A::foo, foo>::ty);`
					`}`
						Add the example (or a similar one) with annotations indicating the intended treatment. Change normative text to produce that treatment as necessary.	

CA 050		06.05		te	In the new paragraph to be added before paragraph 6.5/8 of the base document: Presumably, an entity that is exported should not	Replace “that is introduced by a non-exported declaration” with “that is not exported”.	

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**
					be given module linkage. If it is possible to declare an exported entity with a non-exported declaration, then the wording results in module linkage in too many cases.		
US 051	2	06.05	1	Te	Add a required indication of global module content at the top of a translation unit (see http://wg21.link/p0713).	Require something like "module;" as the first token after preprocessing (i.e., ignoring comments and whitespace).	
US 052		06.05	6	ge	It is surprising that an external-linkage entity first declared at block scope is owned by the global module even if the declaration appears within the purview of a named module and even though the entity may become a member of a namespace contained entirely by a named module.	Give the entity module linkage (just as if the declaration had appeared in the namespace and was not exported).	
US 053		06.05	8	te	Since "the purview of a module" includes the global module, this grants everything module linkage.	Specify "purview of a named module".	
US 054		06.05	8	te	This contradicts /4 by giving names that were already given their (normal) namespace's external linkage (as well as, technically, namespaces not explicitly exported) module linkage instead.	Alter /4 to avoid giving external linkage to module members. Rephrase /8 in terms of "exported" rather than "non-exported declaration".	
US 055		06.05 [basic.link]	2	Te	The set of modules units in a module M is essentially an open set, as new module units can be created at any time. It is not clear how a module unit can look into all other module units to determine if a name is available through module linkage.	Provide a means to close the set of module units that comprise a module.	
US 056		10		ed	The colons for the new productions aren't green.	Make them green!	

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:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
US 057		10		ed	The grammar for module-import-declaration is mis-indented.	Align it with the others.	
US 058		10		ge	Needlessly, a template-declaration can contain an export-declaration: "template<class T> export int i;".	Introduce a grammar production to prevent this (which could also remove the need for explicitly prohibiting "export export int I;").	
US 059		10.01.2		te	In the added paragraph "7": The definition of "owning module" in [dcl.module] relates a module to a declaration, and does not relate a module to an entity. The use of "owning module" in the subject paragraph requires the latter. The statement regarding how an entity is "owned" by a module in [basic.link] appears to apply only to non-exported entities.	Provide a suitable definition for "owning module" or replace its use here.	
CA 060		10.01.2		te	In the added paragraph "7": The definition of "owning module" in [dcl.module] relates a module to a declaration, and does not relate a module to an entity. The use of "owning module" in the subject paragraph requires the latter. The statement regarding how an entity is "owned" by a module in [basic.link] appears to apply only to non-exported entities.	Provide a suitable definition for "owning module" or replace its use here.	
US 061		10.01.2	7	te	Why do we need to specify that a single function has the same address in each translation unit? There is already only one function (without the ODR's help). (If we did need to, it would be wrong to restrict it to the ones importing the module and leave out the module implementation units.)	Remove the stipulation.	

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:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
GB 062		10.01.2	7	Ed	[dcl.fct.spec]p7 second sentence should be a note The second sentence "An exported inline function has the same address in each translation unit[…]" is a special case of the general rule that an inline function has the same address in every translation unit.	Change the second sentence to a note.	
GB 063		10.01.2	7	Te	[dcl.fct.spec]p7 first sentence conflicts with inline function rules [dcl.inline]p6 says "An inline function or variable shall be defined in every translation unit in which it is odr-used", meaning that it is not possible to use an exported inline function from an importing translation unit.	Add changes to p6 excepting this case from the normal rule.	
CA 064		10.03		ed	In the editing instruction at the end of the subject subclause, "10.7" is referred to as a "section" where it may be categorized in a better manner as a subclause. The "as follows" is also odd across a subclause boundary.	Replace "section" with "subclause". Replace "as follows:" with "with the content in subclause 10.7 of this document".	
US 065		10.03	1	ge	[basic.link]/9 prohibits a namespace (with external linkage, as most have since they are automatically exported) in a module implementation unit from sharing a name with (say) a function in another translation unit.	Rename non-exported entities defined in a module implementation unit to avoid the collision.	
GB 066		10.03	1	Te	Not all namespaces should be exported Part of the purpose of the Modules TS is to permit stronger encapsulation. Implicitly exporting all namespaces violates this purpose and should be reconsidered.	Export a namespace only if it is either declared within an export-declaration or contains an export-declaration.	
US 067		10.03	3	te	The statement in terms of grammar productions prevents the auto-export in a namespace from being recursive.	Write "Declarations in an exported namespace-definition are implicitly exported.".	
GB 068		10.07	4	Te	Do modules own namespaces? "A namespace-scope declaration D of an entity	Change the exclusion to "(other than a module or a namespace)"	

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
					(other than a module) in the purview of a module M is said to be owned by M". Should this only apply to declarations that *define* entities? A namespace is unusual in that it can be split over several TUs (and its declarations are also a kind of definition) ; should namespaces be added to the exclusion list?	Note: exclusion wording also needs changing if the change to remove modules from the list of entities is accepted.	
US 069		10.07.1	1	ed	The definition of "interface" is out of place among the constraints.	Put the sentences beginning "The interface of...", "The names of...", and "All entities with..." in a new paragraph.	
US 070		10.07.1	1	ge	Entities cannot be in the interface of a module, since the interface is a set of declarations, not entities.	Write "The names introduced in the interface of a module...".	
US 071		10.07.1	1	te	It is left implied that an export-declaration has the effect of its contained declaration(s).	Say so.	
US 072		10.07.1	1	ge	"export struct A; struct A {}; export struct A;" is said to export the class definition iff the last declaration is present, but no rule establishes completeness as an attribute that can be exported or not.	Add after "importing that module" text describing what the visible names designate (and how that depends on the placement of export-declarations).	
US 073		10.07.1	1	te	The phrase "entities with linkage other than internal linkage" is incorrect, since names are what have (that sort of) linkage.	Rephrase in terms of name visibility (as in the previous sentence).	
GB 074		10.07.1	1	Te	Interface of a module does not contain entities. The interface of a module is defined to be a set of	Modify the wording to make clear which entities are exported by an export-declaration.	

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**
					export-declarations, which is not a set of entities. But paragraph 1 goes on to talk about "entities in the interface of a module", which leaves it unclear precisely which entities are being discussed.		
US 075		10.07.1	2	te	The phrase "types with external linkage" is wrong: a type merely has linkage or not.	Rephrase in terms of the types' names (for linkage purposes).	

GB 076		10.07.1	2	Te	Type restrictions on exported declarations are overly strict.
					The type of an exported declaration is required to only involve types with external linkage. That disallows types without linkage, such as closure types and local types, disallowing in practice many uses of deduced return types. Example:
					error, cannot be exported because return type has no linkage
					`export auto f() {`
					`    return [] { … };`
					`}`
						Delete the type restriction in paragraph 2.	

US 077		10.07.1	paragraph 1	te	Namespaces with external linkage that are exported only by virtue of [basic.namespace] are not specified to form part of the interface of the module from which it is exported.	Insert “all *namespace-definition*s excluding the *namespace-body* and” before “all *export-declaration*s”.	
US 078		10.07.1	paragraph 1	te	The statement regarding names being visible should follow from the specification of [basic.scope] and [basic.lookup]. The statement here is not precise, and has the character of being a candidate for a note. The statement regarding entities being visible should instead deal in names.	Have [basic.scope] and [basic.lookup] contain all of the normative wording regarding names being visible in relation to module units and translation units importing a module. Use a note to cross reference the appropriate subclauses from the subject paragraph.	
CA		10.07.1	paragraph 1	te	Namespaces with external linkage that are exported only by virtue of [basic.namespace] are	Insert “all *namespace-definition*s excluding the *namespace-body* and” before “all *export-*	

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
079					not specified to form part of the interface of the module from which it is exported.	*declaration*s”.	
CA 080		10.07.1	paragraph 1	te	The statement regarding names being visible should follow from the specification of [basic.scope] and [basic.lookup]. The statement here is not precise, and has the character of being a candidate for a note. The statement regarding entities being visible should instead deal in names.	Have [basic.scope] and [basic.lookup] contain all of the normative wording regarding names being visible in relation to module units and translation units importing a module. Use a note to cross reference the appropriate subclauses from the subject paragraph.	
US 081		10.07.1	paragraph 2	te	Presumably, the requirement for external linkage does not apply to “[e]very name” introduced by an *export-declaration*. Instead the requirement applies to names introduced by the *declaration* or *declaration-seq* of an *export-declaration* as interpreted through Clause 10 [dcl.dcl] paragraph 4 of N4660 (with further refinement to allow enumerators, which have no linkage, of unscoped enumerations with linkage). That is, `export auto f(int x) -> decltype(x);` is allowed instead of being ill-formed from the presence of “x” and tenuous applicability of Clause 10 paragraph 4.	Replace “Every name introduced by an *export-declaration*” with “Names introduced by the *declaration* or *declaration-seq* of an *export-declaration*”. Immediately after “external linkage” insert “, or be the name of an enumerator”. Replace “an entity” with “such a name for an entity”.	
CA 082		10.07.1	paragraph 2	te	Presumably, the requirement for external linkage does not apply to “[e]very name” introduced by an *export-declaration*. Instead the requirement applies to names introduced by the *declaration* or *declaration-seq* of an *export-declaration* as interpreted through Clause 10 [dcl.dcl] paragraph 4 of N4660 (with further refinement to allow enumerators, which have no linkage, of unscoped enumerations with linkage). That is, `export auto f(int x) -> decltype(x);` is allowed instead of being ill-formed from the presence of “x” and tenuous applicability of Clause 10 paragraph 4.	Replace “Every name introduced by an *export-declaration*” with “Names introduced by the *declaration* or *declaration-seq* of an *export-declaration*”. Immediately after “external linkage” insert “, or be the name of an enumerator”. Replace “an entity” with “such a name for an entity”.	

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

Date:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
US 083		10.07.1 [dcl.module.interface]		Ed	Exports define the interface of a module, but there doesn't seem to be anything prohibiting export in a module implementation unit.	Change "An export-declaration shall only appear in the purview of a module unit" to "An export-declaration shall only appear in the purview of a module interface unit"	
US 084		10.07.1 [dcl.module.interface]		Te	If a class declaration is exported from an interface module, are the members exported from the class definition found in an implementation module? It is not clear if this is supported, and is essential behaviour for a phased adoption of modules, retaining a traditional #include interface in parallel.	Clarify if necessary, or add missing specification. Add an example to make intent of the final spec clear on this matter, even if other changes are rejected.	

US 085		10.07.1 [dcl.module.interface]		Te	If an implementation module can provide an exported class's definition, how can we export friend functions defined inside a class template, whose signature cannot otherwise be written?	Add an example for the export of `swap` in this template:
						
						`template <class T>`
						`class Wrap {`
						`   T data;`
						
						`  friend void swap(Wrap& lhs, Wrap&`
						`rhs) {`
						`    using namespace std;`
						`    swap(lhs.data, rhs.data);`
						`  }`
						`};`	

US 086		10.07.1, 6.2	4, 6.7	ge	These two paragraphs say that modules own declarations, but 6.4.2/4.4, 6.5/2.2, 6.5/6, and 17.7/7 and /8 all refer instead to entities being owned by modules.	Standardize on declarations; there is no significance attached to the ownership of entities. Alternatively, drop the "own" word entirely in favor of purview.	
US 087		10.07.2	paragraph 1	te	The statement regarding making the exported declarations visible to name lookup should be a note referring to [basic.lookup] or subclauses thereof.	Have [basic.scope] and [basic.lookup] contain all of the normative wording regarding names being visible in relation to module units and translation units importing a module. Use a note to cross reference the appropriate subclauses from the subject paragraph.	
CA		10.07.2	paragraph 1	te	The statement regarding making the exported declarations visible to name lookup should be a	Have [basic.scope] and [basic.lookup] contain all of the normative wording regarding names being	

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:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
088					note referring to [basic.lookup] or subclauses thereof.	visible in relation to module units and translation units importing a module. Use a note to cross reference the appropriate subclauses from the subject paragraph.	
US 089		10.07.3	1	te	"export module B; export import A;" does not make the names from A exported names of B, so a further "export import B;" will fail to propagate them.	Rephrase in terms of altering the export set of B to supply the expected transitivity.	
CH 090		10.07.3	1	ed	Module names in the document are inconsistent, especially using "M'" in this paragraph can be confusing.	s/M'/M1/ or similar pronounceable and clearly distinguishable name for the two mentioned modules. Other places could be affected as well	
US 091		10.07.4	1	te	This paragraph does not say what a proclaimed ownership declaration is for—that is, why it would be used—or what effect it has on clients of the module. (Note: this feature is not mentioned in the referenced proposal.)	Add such an explanation. A commented example would also be helpful.	
US 092		10.07.4	paragraph 2	te	There is no established “owning module” “in” a *proclaimed-ownership-declaration*.	Replace “owning module” with “nominated module”.	
CA 093		10.07.4	paragraph 2	te	There is no established “owning module” “in” a *proclaimed-ownership-declaration*.	Replace “owning module” with “nominated module”.	
US 094 75		17.06.4		ed	In the examples being added in this subclause, the purported module interface units do not contain a *module-declaration* with the `export` keyword. This does not match the definition of module interface units from [dcl.module].	Add the `export` keyword in the appropriate place to the *module-declaration* in each intended module interface unit in the examples.	

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
US 095		17.06.4		ed	In the editing instruction, “a new paragraphs” does not read like proper English.	Replace “a new paragraphs” with “new paragraphs”.	
GB 096		17.06.4		Ed	Missing 'export' in some examples. The examples in 17.6.4 are missing 'export' for the module-declarations in the module interface units.	Prepend "export " to the module declarations for each of F, M, A, B, and C.	
CA 097		17.06.4		ed	In the examples being added in this subclause, one of the primary aspects to consider is the point-of-instantiation of the enclosing template. The examples do not describe the reasoning for why their respective lookups of interest do or do not succeed.	Change the examples to add an indication of the relevant points-of-instantiation for the instantiation context.	
CA 098		17.06.4		ed	The second example being added in the subject subclause seems to imply that the impact of the lookup rules leads to a reliable diagnostic as opposed to cases with no diagnostic required or undefined behaviour from provisions in 17.6.4.1 [temp.point]/8 and 17.6.4.2 [temp.dep.candidate]/1 of WG 21 N 4660.	Provide an example where the lookup rules lead to undefined behaviour due to having, in addition to a viable candidate that would be found in either case, a better candidate that would only be found if the lookup rules were modified to take additional instantiation context into account.	
CA 099		17.06.4		ed	In the examples being added in this subclause, the purported module interface units do not contain a *module-declaration* with the `export` keyword. This does not match the definition of module interface units from [dcl.module].	Add the `export` keyword in the appropriate place to the *module-declaration* in each intended module interface unit in the examples.	
US 100	1	17.06.4	3	Te	The TS should not be issued until a means of writing the currently ill-formed **example**, without requiring the header file to be modified.	Clarify that the lookup context includes the names visible in the set of modules from which types used in the instantiation originated.	
US 101		17.06.4 Temp.dep.res	3	te	This example illustrates a problem, but doesn't show the seriousness of it. The Modules TS support for legacy header does not work. Consider this code:	The Modules TS must require implementations to provide a solution that allows examples such as the above to provide the obvious intended behavior. For example, an implementation could track	

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

Date:2017-10-05	Document:	Project:

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

					foo.h:
					struct A {};
					std::ostream &operator<<(std::ostream&, A);
					namespace N {
					struct B : A {
					  // This could be a friend or a non-member function
					  // in namespace N
					  friend std::ostream &operator<<(std::ostream&, B);
					};
					}
					
					bar.cppm:
					#include "foo.h"
					module bar;
					template<typename T> struct Y {
					  struct Z { N::B b; } z;
					  void f() { std::cout << z.b; }
					};
					#ifndef NDEBUG
					void dump(Y<int> y) { y.f(); }
					#endif
					
					baz.cpp:
					import bar;
					int main() {
					  Y<int> y;
					  y.f();
					}
					
					For a release build (when NDEBUG is defined),
						unresolved argument dependent lookups in templates in a module interface, and ensure that any function that they could select is emitted.
						
						Note that this is also fully addressed by the changes proposed in http://wg21.link/p0273 and http://wg21.link/p0529.	

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

Date:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					this example silently does the wrong thing: only the A base class of the B object is printed. The reason is that foo.h is visible to unqualified lookup (which finds the operator<< for A), but not visible to argument-dependent lookup when instantiating Y<int>::f (so operator<< for B can't be found). Worse, for a debug build (when NDEBUG is not defined), the instantiation context of Y<int>::f() changes to be module bar, changing the behavior of the program. So the bug does not appear when debugging! The proposed TS, in the paragraph cited, describes this issue as an open question. We think this is insufficient.		
US 102		17.06.4	3	ge	There are known cases where this ADL failure changes the meaning of a program instead of making it ill-formed.	Consider carefully whether it is unreasonably expensive to provide the intuitively correct behavior. If it is, provide that (more damaging) example to make it clear for TS users how dangerous the behavior is.	
GB 103		17.06.4	3	Ed	Duplication of 'current' "This example is currently ill-formed by the current specification."	Strike 'currently'	

US 104		17.06.4 [temp.dep.res]	2 and 3	Ed	Module interface units have export in the module-declaration, but the examples do not.	Change for example,
						`module F`
						to
						`export module F`	

US 105		17.06.4 [temp.dep.res]	3	Te	It seems to be reasonable to assume this situation happens fairly often for generic library code. The requirement for having the operators	Make the example well-formed	

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

Date:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					for all the types the template will be instantiated with available at the point of definition of the template seems to hinder the generality of the definition.		
US 106		17.07		ed	Either the editing instruction is unclear as to where the paragraphs are to be inserted, or the paragraph numbering does not reflect the numbering in WG 21 N 4660. There is an existing paragraph numbered as 7 in N4660; the PDTS identifies a new paragraph to be numbered as 7.	Renumber the paragraphs.	
US 107		17.07	all	te	Template specializations do not have module or external linkage (types can either have linkage or not, but that is all).	If there is any way to name the "hidden" specialization from outside the module, prohibit doing so explicitly instead of invoking linkage	
US 108		17.07	all	te	The module ownership of template specializations has no effect (they are not multiply defined (6.2/6.7) and cannot be found by name lookup (6.4.2/4.4)).	Drop the specification of the ownership.	
US 109		2 [intro.refs]	1	Ge	Normative reference should be to a published standard. N4660 is a working draft.	Replace reference to N4660 with a reference to ISO 14882:2017	
US 110		4 [intro]		Ge	Address the third bullet of the committee design principles proposal, P0559R0: “When putting out a TS, a list of questions should be prepared that need to be answered before merging the TS into the C++ working paper. When the questions have been answered, the effort to merge the TS into the C++ WP should get high	Insert a new subclause within 4 General [intro] containing the design questions we actively want feedback on. In particular, how well does this model reflect the concerns for Business Requirements for Modules, P0678R0.	

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

Date:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					priority".		
US 111		6 [basic]	3	Te	Modules need to be able to export typedef names for aliases to entities they do not own, in order to be support purely additive adoption and deployment, so typedef-names must also be entities.	Add *typedef-name* to the list of entities, and alias templates.	
US 112		6 [basic]	3	Te	It is not clear how deduction guides interact with modules, as they are non-members, part of a class interface, but not entities.	Deduction guides should be exported alongside an exported class from the owning module. A module unit should not be allowed to add deduction guides for a class that it does not own.	
US 113		6 [basic]	3	Te	Namespace aliases cannot be exported from an exported namespace within a module, yet are clearly an intended part of the interface.	Add a means to export namespace aliases.	
US 114		6/3		ed	"or this" is not in N4660	Update text being changed.	
US 115		6/3		ge	It does not seem useful to make modules be entities; name lookup cannot find module names, and the rules about declarations, definitions, and linkage do nothing useful for modules.	Strike all changes to 6, 6.1, and 6.3.2 and the sentence about name lookup in 10.7/1.	
US 116		all		ge	We do not believe that the Modules TS in its current form addresses specific, serious aspects of the domain in which it exists: modularization, exporting, and importing interfaces using a semantic model rather than textual inclusion. Specifically, it does not adequately enable modules to be written which use and rely on non-modular code. These limitations extend to both	Examine, refine, and eventually adopt changes such as those proposed by http://wg21.link/p0273 and http://wg21.link/p0529 to address #1, #2, and #3. (The mentioned papers include other changes as well, but they are easily separated.)	

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
					exported interfaces and internal implementation details. We believe there are and will remain substantial bodies of C++ code in shared libraries which do not use modules -- for a variety of reasons -- for a very long period of time (if not forever). This may be required because some portion of users are using older compilers, or merely because the library is no longer being updated. Regardless, this state will be commonplace and pervasive: it is entirely analogous to the situation of many C libraries, which remain to this day a fundamental part of most real world C++ applications. Given this, we think it is critical for a modules system to ensure that these libraries will be able to be used both in modular and non-modular builds. The following use cases show how the current Modules TS falls short of our needs: 1) Use of non-modular library types within the implementation details of a templated interface exported by a module. Currently, the TS does not specify ADL rules adequate to ensure users of an exported templated interface would have consistent and expected behavior (see the next comment). Potential workarounds involve untenable approaches, such as requiring all users to #include an implementation detail header when importing a module. Another potential workaround involves using novel and surprising using declarations to enumerate the details of the non-modular library in a way that triggers re-export. However, this would require the author of the module to know the exact implementation details of the non-modular library they are using and encode them in their usage. Any change to the non-modular interface would require updates in all such users which, again, seems untenable.		

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
					2) Authoring a library whose API is made available both through a header and a module, and where the result is both syntax and ABI compatible. This requires a rich ability to export entities whose one definition is in a non-modular library through a modular interface. The proposed mechanisms for this are extremely cumbersome and impose severe functional limitations. The approach also needs to be viable for *existing libraries* to adopt at scale, which we see as a requirement for modules themselves to be adopted at scale. This means that a usable approach must not presuppose a dramatic redesign or reorganization of the APIs or header files used by existing libraries. The approach also must support fundamental API facilities in widespread use, even if distasteful, such as macros. Without this, adoption of the modules system will be fragmented and slow, and we believe it will ultimately not achieve its goals. 3) A non-modular existing library which exposes its API through a module with the mechanisms of #2 must be able to *incrementally* move its API into a fully modular form without breaking user code or changing ABI. Again, for us this is an essential property of a C++ modules system which we expect to see widespread adoption. Consider the process of incrementally adopting modules across an existing, complex C++ codebase. One approach would be to modularize "top down", or from the leaves of the project. However, following this approach results in problematic, buggy behavior due to the issues in #1. Another approach would be "bottom up", or from the lowest level components. However, the lowest level component available will almost never be the actual root. More often, it will depend on C libraries and other non-modular code that cannot		

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:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					be converted (either by lack of control, or by explicit exclusion under the Modules TS -- such as C libraries). Consequently, the modular "roots" would require re-export of macros and other state to preclude ODR violations when non-modular code is exported transitively through modular code, and back into other, non-modular code. The end result is that there is no realistic incremental adoption strategy for large existing codebases. Instead, modules will only be usable when starting from new components, in a new system. A modules system lacking any of these facilities and unable to be incrementally adopted at scale will not be widely usable for our users. But we do have a pressing need for solutions to the problems that a modules system provides, we know about the above issues, we know how to address them, and have both a proposal addressing them and extensive real-world implementation experience with that approach. This different approach is something we can adopt, and we suspect other users of C++ will adopt it as well. Our views and objections have been shaped by iteratively developing and deploying a modular model very close to the current TS. Early in that effort, we discovered that these very low-level minutiae are critical for widespread adoption. Ignoring these considerations seems unwise for such a substantial change to the C++ language -- especially as these objections are borne from practical experience. Beyond not being useful for addressing our needs of a modules system in C++, we feel that publishing the modules TS as-is, without addressing these already known issues, will cause fragmentation in the C++ ecosystem. Different projects will adopt different systems, and the 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

Date:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					for users will be having to reason about two, different modular systems. Given the ability to avoid this confusion and the fact that we already have strong understanding of these issues, we feel that publishing the TS as is would harm the community without providing significant benefit.		
US 117		ALL		ge	A module system without support for macros is unshippable in our ecosystem.	Add support for macros.	
US 118		General		ed	The List of Tables contains no content.	Remove the List of Tables.	
US 119		General		ed	The document presented does not meet the requirement in subclause 22.3.1 of the ISO/IEC Directives, Part 2 that the clause numbering shall be continuous: Clause 10 immediately follows Clause 6, and Clause 17 immediately follows Clause 10.	Renumber or add intervening Clauses.	
US 120		General		ed	The word “section” (and its plural form) appears in various places of the PDTS document where the corresponding form of “subclause” is probably meant.	Replace “section” with “subclause” as appropriate throughout the document. Do the same for the corresponding plural forms.	
CA 121		General		ge	Further elaboration over the role of the module interface unit would be helpful.	Either add notes through the editing instructions as text to be applied to the base document, or make recommendations on the use of module interface units in an informative annex.	

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:2017-10-05	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
CA 122		General		ge	An exploration of possible implementation strategies and models—considering what sort of artifacts, extra payload, and configuration may be necessary in the build environment at various stages—would provide helpful information for implementors and users alike.	Add an informative annex explaining various models. Indicate in the informative annex how each model would have different implications on the subclauses labelled [lex.separate] and [lex.phases] in WG 21 N 4660.	
CA 123		General		te	Presumably in phase 8 of translation (from WG 21 N 4660 subclause 5.2 [lex.phases]), it is not meant for it to be implementation-defined whether or not the source of the translation units containing the definitions of exported templates is required should an instantiation be necessary. Alternatively, it is probably meant for it to be implementation defined whether or not the source of module interface units, module units in general, or translation units importing modules is required to be available in phases 7 and 8.	Modify [lex.separate] and [lex.phases] to clarify what, if any, source is intended to be required at different stages of translation.	
CA 124		General		te	It is unclear from the PDTS what the notion of a global module is intended to achieve. The PDTS wording produces surprising results in various places when referring to the "purview of a module" as opposed to the "purview of a *named* module" in [basic.def.odr] (prohibiting multiple definitions entirely) and in [basic.link] (granting module linkage to many names).	Consider reducing the applicability of individual rules to apply only to named modules (explicitly noting cases where the global module is included). Alternatively, reduce the purview of the global module by requiring opt-in at the source level.	
CA 125		General		te	It is presumably not intended for `main` to be possibly owned by a named module and not exported.	Modify [basic.start.main] as necessary.	
CA 126		General		ed	In subclause C.2.7 of WG 21 N 4660, it is documented that `export` has been removed from C++. The PDTS restores `export` in some form.	Add an editing instruction to update the subject subclause.	
US 127		General	05.2	te	Presumably in phase 8 of translation (from WG 21 N 4660 subclause 5.2 [lex.phases]), it is not meant	Modify [lex.separate] and [lex.phases] to clarify what, if any, source is intended to be required at	

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
					for it to be implementation-defined whether or not the source of the translation units containing the definitions of exported templates is required should an instantiation be necessary. Alternatively, it is probably meant for it to be implementation defined whether or not the source of module interface units, module units in general, or translation units importing modules is required to be available in phases 7 and 8.	different stages of translation.	
US 128		General	ALL	ge	Further elaboration over the role of the module interface unit would be helpful.	Either add notes through the editing instructions as text to be applied to the base document, or make recommendations on the use of module interface units in an informative annex.	
US 129		General	ALL	ge	An exploration of possible implementation strategies and models—considering what sort of artifacts, extra payload, and configuration may be necessary in the build environment at various stages—would provide helpful information for implementors and users alike.	Add an informative annex explaining various models. Indicate in the informative annex how each model would have different implications on the subclauses labelled [lex.separate] and [lex.phases] in WG 21 N 4660.	

```
D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 21544 - JTC001-SC22-N5233_ANSI.docx: Collation successful

D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 21544 - JTC001-SC22-N5233_BSI.doc: Collation successful

D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 21544 - JTC001-SC22-N5233_JISC.doc: Collation successful

D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 21544 - JTC001-SC22-N5233_SCC.doc: Collation successful

D:\ISO\data\prod_iso_comment-collation\work\temp\ISO_IEC PDTS 21544 - JTC001-SC22-N5233_SNV.doc: Collation successful

Collation of files was successful. Number of collated files: 5

SELECTED        (number of files):  5
```

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:2017-10-05	Document:	Project:

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

PASSED TEST (number of files): 5

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