Doc No: SC22/WG21/N1760
J16/05-0020
Date: Nov 10, 2004
Project: JTC1.22.32
Reply to: Herb Sutter
Microsoft Corp.
1 Microsoft Way
Redmond WA USA 98052
Email: hsutter@microsoft.com

TG5 Liaison Report #7

Meeting #8 of Ecma TC39/TG5 (C++/CLI) was held in Redmond, WA, USA, on October 22–23, 2004.

The following TG5 documents are attached to this liaison report:

- TC39-TG5/2004/41 Agenda for the 8th meeting of TC39-TG5, Redmond, October 2004
- TC39-TG5/2004/42 **Intentionally omitted (see below)**
- TC39-TG5/2004/43 Project Editor's Report, October 2004
- TC39-TG5/2004/44 C++/CLI Specification Comments - revision 14 October 2004
- TC39-TG5/2004/45 C++/CLI Specification Comments - revision 26 October 2004
- TC39-TG5/2004/46 Minutes of the 8th meeting of TC39-TG5, Redmond, October 2004

Document TC39-TG5/2004/42, "Working Draft 1.8 of the C++/CLI Standard, Language" is not included. This draft can be found at the following URLs:

- http://www.plumhall.com/ecma/index.html
- http://msdn.microsoft.com/visualc/homepageheadlines/ecma/default.aspx
- http://www.dinkumware.com

Agenda for the: 8th meeting of Ecma TC39-TG5

to be held in: Redmond, WA, USA

on: 22-23 October 2004

TIME: **13:00!! till 17:00 on Fri 22nd October 2004**
09:00 till 17:00 on Sat 23rd October 2004
[Noon lunch each day]

LOCATION: **Fri 22nd October: Bldg 44, Room 3200**
Sat 23rd October: Bldg 41, Room 2731
Microsoft Campus, Redmond WA 98052 USA
(Directions: see TG5/2004/021)

CONTACT: **John Hawkins**
johawk@microsoft.com

1 Opening

1.1 Appointment of Recording Secretary
1.2 Introduction of participants
1.3 Host facilities/local information

2 Adoption of the agenda

3 Final approval of minutes of previous TG5 meeting (2004TG5 040)

4 Matters arising from the minutes not covered elsewhere

5 Project Editor's Report

6 Approving tracked changes in latest draft

7 Date and place of next meetings

7.1 January xxx, Westfield, NJ; hosted by EDG/Dinkumware
7.2 March 8, 9, and 11(9am), Kona, HI; hosted by Plum Hall

NOTE
TC39 business meeting takes place March 11(pm)

8 Reports from Liaisons

8.1 TC39 TG3 (CLI) – Rex Jaeschke

8.2 SC22/WG21 (C++) – Tom Plum, P. J. Plauger, Tana Plauger, John Spicer, and Steve Adamczyk

8.2.1 explicit conversion functions (#105, Hall)

8.3 TC39 TG2 (C#) – Rex Jaeschke

9 Action item spreadsheet review

10 Any other business, and appreciation of hosts

11 Adjournment

This is a replacement/place-holder for Document TC39-TG5/2004/42, “Working Draft 1.8 of the C++/CLI Standard, Language”. This draft can be found at the following URLs:

- http://www.plumhall.com/ecma/index.html
- http://msdn.microsoft.com/visualc/homepageheadlines/ecma/default.aspx
- http://www.dinkumware.com

2004-10 Project Editor's Report

Rex Jaeschke

ECMA TC39-TG5 project editor

rex@RexJaeschke.com

+1 703 860-0091

Working Draft 1.8 has been produced and distributed. The following work went into producing it:

1. I applied corrections resulting from the Redmond Sep meeting.
2. I created Clause 33, "Metadata", and moved all metadata-related text from the other clauses (except for the custom modifier text in Clause 32) to this new clause. Only new metadata in this clause has been tracked.
3. I made many improvements of an editorial nature; these were not tracked. (They included changing numerous examples so that properties were no longer in native classes.)
4. I merged in Steve's long long proposal.
5. Some issues I came across while working on the draft:
    a. In email on Oct 6, I raised the question of whether a ref class can be implicitly abstract if it contains an abstract or pure virtual method.
    b. 18.9, "Static constructors", states: "A static constructor can have any *access-specifier*. [*Note:* However, for security reasons, a static constructor should have a `private` *access-specifier*." True, it can have any access-specifier; however, the compiler always emits it in metadata as `private`. If that is correct, should we allow it to de declared with any access-specifier other than `private`?

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
10	4-Dec-03	meeting #1 (TX)	14	Technical	M	Brandon Bray	pull together all the conversion information into one place. Make sure all conversions are covered.		No	
11	4-Dec-03	meeting #1 (TX)	15.3.2	Technical		Steve Adamczyk	comma vs. semicolon as separator in indexed access expressions In indexed access expressions (§15.3.2), comma operators are currently disallowed inside [ ] unless they are enclosed in parentheses. This conflicts with usage in existing template libraries (e.g., Lambda), in which the comma operator occurs inside [ ] without enclosing it in parentheses.	Meeting #2 (HI): Can we treat commas in [ ] not having enclosing parenthesis, in any context, always be treated as punctuators? Yes. Steve will provide words to the editor for this. Meeting #3 (Mel): Steve produced a paper. He reported one outstanding issue: In 15.3.2, "Indexed Access", in the C++/CLI spec is rather vague. There, we have indexed-access: indexed-designator [expression-list] where indexed-access is defined as an additional alternative for postfix-expression: postfix-expression: indexed-access Unfortunately, there isn't any definition of indexed-designator, so I'm not quite sure whether all the multi-dimensional cases are supposed be handled by indexed-designator, leaving the traditional cases to be handled by the original (possibily modified) syntax. An alternative would be not to introduce indexed-access at all, and use the definition postfix-expression: postfix-expression [expression-list] to handle all the cases, for both traditional subscripting and the new C++/CLI indexer references. There was agreement to this, so Steve will update his p	No	
13	4-Dec-03	meeting #1 (TX)	12	Technical	M	Brandon Bray	Add a diagram of the type tree		No	
19	5-Dec-03	meeting #1 (TX)		Technical	L	Brandon Bray	list of overlap between Standard C++ and features proposed by C++/CLI		No	
23	16-Dec-03	Phone meeting	8.2.3	Editorial	H	Brandon Bray	Say more, especially w.r.t the template class array<element-type>.		No	
24	16-Dec-03	Phone meeting	9	Technical	R	Brandon Bray	Review this clause.		No	
25	16-Dec-03	Phone meeting	10	Technical	H	Brandon Bray	Revise this clause by covering topics including application entry point, assembly boundaries, among others.		No	
27	16-Dec-03	Phone meeting	12.13.6	Technical	H	Brandon Bray	Describe how interior_ptr, pin_ptr, array, and safe_cast are template-like with certain constraints.		No	
28	16-Dec-03	Phone meeting	12.3.6	Technical	M	Brandon Bray	Describe how the compiler will need to emit a modopt to distinguish interior_ptr<T> from tracking reference to T (T%) in the metatada.		No	
29	16-Dec-03	Phone meeting	12.3.6.2	Technical	M	Brandon Bray	Spell out target type restrictions		No	
32	16-Dec-03	Phone meeting	13	Technical		Tom Plum	What, if anything, goes in this clause?		No	
33	16-Dec-03	Phone meeting	14.1.1	Editorial	R	Brandon Bray	Review this subclause.		No	
34	16-Dec-03	Phone meeting	14.4	Editorial	R	Brandon Bray	Review this subclause.		No	
35	16-Dec-03	Phone meeting	15.1	Technical	H	Brandon Bray	The rewrite rules for e[x] (default indexed accesses) are different where there is only one index. This is because there is a potential ambiguity with the C++ operator[]. Is this mentioned elsewhere?		No	
36	16-Dec-03	Phone meeting	15.3.8	Technical	M	Brandon Bray	cv-qualification needs to be considered.		No	
38	16-Dec-03	Phone meeting	15.3.9	Technical	L	Brandon Bray	Provide a spec for standard typeid (that returns std::type_info) in addition to the new typeid (that returns System::Type).		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
39	16-Dec-03	Phone meeting	15.3.13	Editorial	H	Brandon Bray	Update this subclause		No	
40	16-Dec-03	Phone meeting	15.4.1.1	Editorial	R	Brandon Bray	Review this subclause.		No	
43	16-Dec-03	Phone meeting	15.11.1	Technical		Mark Hall	Add support for handle equality comparison, and handle ==/!= nullptr, and vice versa.	Meeting #3 (Mel): Had a short discussion. Mark will produce a paper for the May meeting. Meeting #4 (NJ): No progress. To be discussed via email, and at the Jun meeting Meeting #5 (WA): Discussed briefly. Asked Mark to write this up and distribute to the reflector. Phone call Jun 29: This issue was resolved; just needs drafting of final words. Meeting 7 (WA): In the case of if(handle), which conversions are attempted before comparison against nullptr is used? We agreed that if an explicit conversion to bool exists, if(handle) uses that. There is no implicit unboxing. Steve and Mark worked on this and presented it to the full committee on the 2nd day. Based on committee feedback, Mark will write this up for future consideration.	No	
44	16-Dec-03	Phone meeting	15.18	Technical	H	Brandon Bray	Add words to discuss assignment for properties and events from the point of view of the rewrite rules.		No	
47	16-Dec-03	Phone meeting	17	Technical	M	Brandon Bray	Provide text for this clause		No	
50	16-Dec-03	Phone meeting	18.4	Technical	M	Brandon Bray	Extend declarator-id's by adding a new production that allows default.		No	
52	16-Dec-03	Phone meeting	18.4.2	Technical	H	Brandon Bray	This subclause only covers how the accessor functions must be defined. The expressions clause needs to cover the rewrite rules that call accessor functions.		No	
54	16-Dec-03	Phone meeting	18.5.2	Editorial	R	Brandon Bray	Review this subclause.		No	
55	16-Dec-03	Phone meeting	18.6	Editorial	R	Brandon Bray	Review this subclause.		No	
56	16-Dec-03	Phone meeting	18.6.4	Technical	M	Brandon Bray	Identify when synthesis would and would not occur.		No	
57	16-Dec-03	Phone meeting	18.6.5.1	Technical	L	Brandon Bray	Writeup op_true and op_false operators		No	
58	16-Dec-03	Phone meeting	18.6.6.1	Technical		Mark Hall	Reword this subclause similarly to the way special member functions are described.	Meeting 7 (WA): ?? To be done in Tue morning work sessions.	No	
59	16-Dec-03	Phone meeting	18.6.6.1	Technical	H	Brandon Bray	Add another subclause to cover the compiler-generated conversion from handle to unspecified bool type.	Meeting 7 (WA): ?? To be done in Tue morning work sessions.	No	
62	16-Dec-03	Phone meeting	18.10.1	Technical	L	Brandon Bray	Add a description that for any value class we have to make the copy before calling member functions.		No	
63	16-Dec-03	Phone meeting	18.11	Technical	H	Brandon Bray	Say more about finalizers (including Dispose/~T and Finalize/!T) and add some examples.		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
65	16-Dec-03	Phone meeting	18.1	Technical		Editor	As a cross-language issue, come up with terminology to distingish between destructors and finalizers. Perhaps "deterministic destructor" vs. "non-deterministic finalizer." Add some text in spec re this, esp. w.r.t C#'s use of destructor.		No	
66	16-Dec-03	Phone meeting	21	Editorial	M	Brandon Bray	Introduce value classes -- Discuss the following: value classes are optimized for small data structures. As such, value classes do not allow inheritance from anything but interface classes. Tie in fundamental classes.		No	
67	16-Dec-03	Phone meeting	21.4.1	Technical	H	Brandon Bray	Add words about instance constructors and static constructor. Value classes cannot have SMFs (specifically, default constructor, copy constructor, assignment operator, destructor, or finalizer. Need to add specification for this along with rationale.		No	
68	16-Dec-03	Phone meeting	22	Technical	L	Brandon Bray	Consider writing some text for this "place-holder" clause. Should this all go in the new annex "Future directions"?		No	
71	16-Dec-03	Phone meeting	23	Editorial	R	Brandon Bray	Will review this whole clause.		No	
74	16-Dec-03	Phone meeting	23.5	Technical	M	Brandon Bray	Look at array covariance w.r.t arrays having copy constructors.		No	
75	16-Dec-03	Phone meeting	23.6	Technical	M	Brandon Bray	Write up array initialization.		No	
76	16-Dec-03	Phone meeting	24.4	Technical	H	Brandon Bray	Address what happens when a ref class does not implement an interface function (and what happens when a base class has a non-virtual function with the same name).		No	
79	16-Dec-03	Phone meeting	27	Technical	H	Brandon Bray	Cover unification of CLI and Standard C++ exception-handling models, and anything else that might go in this clause. Are exceptions asynchronous now in some cases? Yes they are. (For example, NullReferenceException.)	Meeting #5 (WA): Kevin Free (Microsoft) gave a verbal presentation. catch(…) catches managed and native exceptions. catch(System::Object^) also catches both kinds, but won't invoke the destructor (so can leak). CLI exception handling supports more features than we expose. The issue remained with Brandon to write up, as before.	No	
81	16-Dec-03	Phone meeting	20.5.2	Technical	R	Brandon Bray	Describe MethodImplOption metadata generation.		No	
82	16-Dec-03	Phone meeting	29	Technical	M	Brandon Bray	Flesh out "Templates" clause.		No	
87	16-Dec-03	Phone meeting	A	Technical	L	Brandon Bray	Flesh out "Verifiable code" clause. Describe the dangers of pointer arithmetic and interior_ptrs.		No	
88	16-Dec-03	Phone meeting	B	Technical	L	Brandon Bray	Flesh out "Documentation comments" clause.		No	
90	16-Dec-03	Phone meeting	D	Technical		Editor	Add naming guidelines for generics		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**

92	29-Jan-04	meeting #2 (HI)		Technical	M	Brandon Bray	"size size" name lookup issue (see email thread started by Herb Sutter on January 14 on the liaison reflector under the topic {Name lookup 1 (of 2): "Size Size" (CLI property naming idiom)}.)
							
							This is the common CLI idiom of naming a property (or potentially other members) with the same name as its type. In particular, here are two common examples:
							
							value class Size { /*…*/ };
							
							value class Color { /*…*/ };
							
							ref class X {
							public:
							property Size Size;
							property Color Color;
							};
							
							In other languages, it's easy to simply use the identifier "Size" without qualification and have the compiler Do the Right Thing™. But C++ name lookup is different. The status quo in Managed C++ syntax was that we made no change to C++ lookup rules, with the result that authors of classes that use this idiom are required to qualify most occurrences of "Size" which is ugly. The issue mostly appears only within the class itself (and in derived classes).
							
							Here's a brief description of the problem:
							
							ref class X {
							public:
							property Size Size {
									No	

94	29-Jan-04	meeting #2 (HI)		Technical		Mark Hall	Relationship between primitive types and CLI types. The current spec allows the following: int i = 10; String^ s = i.ToString(); Standard C++ doesn't allow member selection on expressions of primitive type. Assuming int maps to System::Int32, just how much alike are these two types? Specifically, when do we treat the primitive as the underlying class.	Meeting 5 (WA): Asked Mark to write this up and distribute to the reflector. Please address the side-effect issue; that is, given (i++).ToString, is the increment done? Meeting 7 (WA): ?? To be done in Tue morning work sessions. Re the side-effect, yes, it must be done.	No	
95	29-Jan-04	meeting #2 (HI)	10	Technical	H	Brandon Bray	Provide words for #using.		No	
96	29-Jan-04	meeting #2 (HI)	9.1.1	Technical	M	Brandon Bray	The spec does not provide a way to use a keyword as an identifier. (Managed C++ used the intrinsic __identifier(name) to achieve this; C# uses a leading @.) This is an issue for inter-operability; for example, being a consumer of a public type (written in something other than C++) that has a name (or contains a public member that has a name) that is a keyword in C++.		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
97	29-Jan-04	meeting #2 (HI)		Technical		Editor	Overloading on arity. (This is a liaison issue with TG3.) The issue involves the overloading of a non-generic type with a one or more generic types of the same name in the same namespace. For example, the following is permitted by the CLS: ref class X { /*…*/ }; generic<typename T> /*…*/ ref class X { /*…*/ }; generic<typename T, typename U> /*…*/ ref class X { /*…*/ };	Meeting 3 (Mel): Herb presented this issue, which was then reassigned to Brandon. Meeting 5 (WA): In this version, we'll support a generic and non-generic version of a type in the same namespace, but not in different namespaces. There was a discussion about using something like "using generic x::y" to provide cross-namespace support as well. Rex to work with Brandon to get this into the draft. Meeting 7 (WA): Herb reported that the MS implementation can consume same-named generics that overload on arity in the same assembly, but it cannot create them.	No	
98	29-Jan-04	meeting #2 (HI)	30	Technical	R	Brandon Bray	Restrictions on generics re generic code generation. The current generics clause needs to be fleshed out, especially w.r.t how overload resolution works within the CLI.	Meeting #2 (HI): Brandon will write a paper on this. Meeting #4 (NJ): The fleshing out of Clause 30 is a significant contribution toward this. More work needed in declarations and function calls.	No	
105	29-Jan-04	meeting #2 (HI)	14.5.1	Technical		Mark Hall	Constructors can't be used in casts in managed classes. Should they be allowed in explicit conversions? All managed type constructors being explicit by default. (Already yes, but reconfirm this.)	Meeting #4 (NJ): Steve will send the editor sufficient text to go into the public drop to indicate our intention re this topic. DONE. Meeting 5 (WA): Asked Mark to write this up and distribute to the reflector. Meeting 7 (WA): Steve and Mark worked on this and presented it to the full committee on the 2nd day. Mark will write this up for future consideration.	No	
106	29-Jan-04	meeting #2 (HI)		Technical		Daveed Vandevoorde	Should >> handled as two tokens rather than one; e.g., List<List<int>>.	Meeting #3 (Mel): Had a short discussion. Tom will produce a paper for the May meeting. Meeting #4 (NJ): TG5 agreed that if a < for a template is seen, and >> that are not inside parentheses, that >> will always be considered to be the closing delimiter of two < symbols, and results in an error if there are not two such corresponding < symbols. Refer to Daveed's paper WG21/N1649 for more information. Meeting #7 (WA): This paper was updated (see N1699). It was discussed in TG5 and will be discussed at the up-coming WG21 meeting, at which TG5 members will participate.	No	
109	19-Feb-04		12.3.6.3	Technical	L	Brandon Bray	Cover the dangers of pointer arithmetic and interior_ptrs		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
111	19-Feb-04		15.3.2	Technical	M	Brandon Bray	Need to consider how indexed access expressions are interpreted in templates.		No	
116	19-Feb-04		18.4.2	Technical	H	Brandon Bray	Add some discussion of how accesses to properties are rewritten into accessor functions. This should be covered in rewrite rules in the expressions clause. Note that access checking for whether a property can be written to or read to is done after rewriting and overload resolutions.		No	
117	19-Feb-04		18.4.2	Technical	H	Brandon Bray	The qualified name of a property needs to be described somewhere. Once that happens, how an out-of-class definition is done will already be covered by existing rules.		No	
121	19-Mar-04	meeting #3 (Mel)		Technical		Steve Adamczyk	In the context of Herb's keywords paper (2004-05), Steve will write up the notion "If it can be an identifier, it is."		No	
122	19-Mar-04	meeting #3 (Mel)		Technical		Steve Adamczyk	Write a WG21 paper on extended integer types, promotion rules, costs of conversion, and the like, for the May meeting.	Meeting #4 (NJ): Not yet done, but still planned.	No	

124	10-Jun-04	Jonathan Caves		Technical		Jonathan Caves	Indexed properties -- Consider the following:
							
							interface class I1 {
							  property int Value;
							};
							
							interface class I2 {
							  property int Value[String^] {
							  int get(String^);
							  void set(String^, int);
							  };
							};
							
							ref class D : I1, I2 {
							  // Implements the properties
							};
							
							D^ d;
							d->Value["Foo"];
							
							The question is what does the last line do?
							
							Which leads to a language design question - what should the complier do when faced with a property followed by a '['
							
							1) Should it look for just parameterized properties and if there isn't one fail - I suspect not
							
							2) Should it look for all properties and if the returned set contains a parameterized property it should prefer it - this sounds like magic to me.
							
							3) Should it look for all properties perform overload resolution across the whole set and it the resulting call is ambiguous then issue an error.
								Meeting #5 (WA): Discussed this. Option #3 preferred.
								
								Meeting 7 (WA): Discussed this in detail.
								
								property int Value[int] {
								  void set(int, int);
								};
								
								x->Value[1] = 4
								is treated as
								x->set_Value(1,4);
								
								--- 	
								
								property array<int>^ Value {
								  array<int>^ get();
								}
								
								x->Value[1] = 4
								is treated as
								x->get_Value()[1] = 4
								
								--- 	
								
								property int% Value[int] {
								  int% get(int);
								}
								
								x->Value[1] = 4
								is treated as
								x->get_Value(1) = 4
								
								This construct violates the principle of properties (that of setting/getting the value of some property), so is not to be encouraged; however, it is supported, but no need to consider it further here.
									No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**

125	14-Jun-04	meeting #5 (WA)	8.15.3	Technical	M	Brandon Bray	Based on the rules for type deduction in templates, it seems surprising that you can match array<ItemType>^ with an argument of type int. Here is a standard C++ example intended to illustrate the issue:
							template <class ItemType> struct Stack {};
							template <class ItemType> struct Array {
							Array(ItemType);
							};
							template <class ItemType>
							void PushMultiple(Stack<ItemType>, Array<ItemType>);
							int main() {
							Stack<int> s;
							PushMultiple(s, 1); // deduction fails
							PushMultiple<int>(s, 1);
							}
							Are the rules for generic different in this area? [There seems to be information related to this in 30.3.2. See that subclause for further comments on this issue.]
									No	

127	14-Jun-04	meeting #5 (WA)	12.3.3	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the modreq IsBoxed shall be emitted (i.e., passing		No	
128	14-Jun-04	meeting #5 (WA)	12.3.6	Technical	L	Brandon Bray	The compiler will need to emit a modopt to distinguish interior_ptr<T> from tracking reference to T (		No	
129	14-Jun-04	meeting #5 (WA)	12.3.7	Technical	L	Brandon Bray	Need to add text to indicate the circumstances under which the modopt IsPinned shall be emitted (i.e.,		No	
130	14-Jun-04	meeting #5 (WA)	14.1.1	Technical	L	Brandon Bray	Separate the list of conversions from the order of preference (such as how Standard C++ separates Sta		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
131	14-Jun-04	meeting #5 (WA)	15.3.3	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the following type modifiers shall be emitted, and point to each modifier's definition: • IsBoxed i.e., passing a handle to a value type). • IsByValue (i.e., ref class type passed by value). • IsConst (i.e., pointer or reference to a const-qualified type). • IsExplicitlyDereferenced (i.e., interior_ptr as a parameter). • IsImplicitlyDereferenced (i.e., parameter is a reference). • IsLong (i.e., long/unsigned long/long double parameters). • IsExplicitlyDereferenced (i.e., pin_ptr as a parameter). • IsSignUnspecifiedByte (i.e., plain char's sigedness). • IsUdtReturn (i.e., ref class type returned by value). • IsVolatile (i.e., pointer or reference to a volatile-qualified type).		No	
132	14-Jun-04	meeting #5 (WA)	15.3.10	Technical	M	Brandon Bray	Unboxing and boxing are described as preferred user-defined conversions. Nothing important about th		No	
133	14-Jun-04	meeting #5 (WA)	15.3.10	Technical	L	Brandon Bray	The null value is converted to the null value of the destination type. This can be unverifiable and migh		No	
134	14-Jun-04	meeting #5 (WA)	16.3.3	Technical	L	Brandon Bray	Need to add text to indicate the circumstances under which the modreq IsUdtReturn shall be emitted (		No	
135	14-Jun-04	meeting #5 (WA)	18	Technical	R	Brandon Bray	This table and corresponding sections should include Special Member Functions (SMFs) like destruct		No	
136	14-Jun-04	meeting #5 (WA)	18.2.1	Technical	L	Brandon Bray	Need to address the following: C++/CLI uses the System::Reflection::DefaultMemberAttribute attribu		No	
138	14-Jun-04	meeting #5 (WA)	18.4	Technical		Mark Hall	Need to write up the restrictions on trivial properties.		No	
139	14-Jun-04	meeting #5 (WA)	18.4	Technical	L	Brandon Bray	We probably should say something about the reserved names get_Item and set_Item, and their relation		No	
142	14-Jun-04	meeting #5 (WA)	18.5.3	Technical	L	Brandon Bray	An event with the new modifier introduces a new event that does not override an event from a base cl		No	
143	14-Jun-04	meeting #5 (WA)	18.6	Technical	L	Brandon Bray	The restriction below does not apply to non-static member operators – that need not have a parameter		No	
144	14-Jun-04	meeting #5 (WA)	18.6.1	Technical	L	Brandon Bray	Provide an example for "Homogenizing the candidate overload set".		No	
145	14-Jun-04	meeting #5 (WA)	18.6.5.2	Technical	L	Brandon Bray	Provide C++ names for operator True and False		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
148	14-Jun-04	meeting #5 (WA)	20.2	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the following type modifiers shall be emitted, and point to each modifier's definition: • IsConst (i.e., data member involves a cv type). • IsImplicitlyDereferenced (i.e., has a reference type). • IsLong (i.e., long/unsigned long/long double type). • IsSignUnspecifiedByte (i.e., plain char's sigedness). • IsVolatile (i.e., data member involves a cv type).		No	
149	14-Jun-04	meeting #5 (WA)	20.3	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the following type modifiers shall be emitted, and point to each modifier's definition: • IsBoxed i.e., passing a handle to a value type). • IsByValue (i.e., ref class type passed by value). • IsConst (i.e., pointer or reference to a const-qualified type). • IsExplicitlyDereferenced (i.e., interior_ptr as a parameter). • IsImplicitlyDereferenced (i.e., parameter is a reference). • IsLong (i.e., long/unsigned long/long double parameters). • IsExplicitlyDereferenced (i.e., pin_ptr as a parameter). • IsSignUnspecifiedByte (i.e., plain char's signedness). • IsUdtReturn (i.e., ref class type returned by value). • IsVolatile (i.e., pointer or reference to a volatile-qualified type).		No	
151	14-Jun-04	meeting #5 (WA)	24.2	Technical	M	Brandon Bray	The note says "pickup the restrictions from page 333". Brandon, do you have any idea what this page		No	
154	14-Jun-04	meeting #5 (WA)	30.1	Technical	R	Brandon Bray	Doesn't the text "a generic name declared in namespace scope or in class scope shall be unique in that		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
155	14-Jun-04	meeting #5 (WA)	30.1	Technical	R	Brandon Bray	What is a non-generic type? Does it mean that the rules are the same as classes? As template classes?		No	
156	14-Jun-04	meeting #5 (WA)	30.1	Technical		Editor	Can generic types be nested in native classes?		No	
158	14-Jun-04	meeting #5 (WA)	30.1.1	Technical	R	Brandon Bray	The equivalent wording for template parameters in the working paper has been changed to "defines its		No	
159	14-Jun-04	meeting #5 (WA)	30.1.2	Technical	R	Brandon Bray	30.1.2 says "Like templates in Standard C++, within the body of a generic type any usage of the unqualified unadorned name of that type is assumed to refer to the current instantiation." 30.1.3 then goes on to describe "The instance type". Those seem like to different ways of describing the same concept. Can they be unified in some way?		No	
160	14-Jun-04	meeting #5 (WA)	30.1.6	Technical	R	Brandon Bray	This subclause describes when a static constructor is invoked. In 18.8, it references the CLI Standard Partition II (10.5.3). Are the rules the same? (Yes) Should this subclause also just reference the CLI spec? There are two sets of behavior; we need to say which one we use.		No	
161	14-Jun-04	meeting #5 (WA)	30.1.7	Technical	M	Brandon Bray	What to say about explicit conversion functions (which can only occur in managed class types)?		No	
162	14-Jun-04	meeting #5 (WA)	30.2.2	Technical	R	Brandon Bray	This subclause lists the types that can and cannot be generic arguments. Fundamental types are not in		No	
163	14-Jun-04	meeting #5 (WA)	30.2.4	Technical	R	Brandon Bray	"The non-inherited members of a constructed type are obtained by substituting, for each generic-parameter in the member declaration, the corresponding generic-argument of the constructed type. The substitution process is based on the semantic meaning of type declarations, and is not simply textual substitution." It would be helpful to explain this in more detail and/or give an example where this makes a difference.		No	
165	14-Jun-04	meeting #5 (WA)	30.3	Technical	L	Brandon Bray	Types not used as a parameter type to a generic function cannot be deduced. Are the nondeduced context rules the same as Standard C++ or not? The sentence before this is true, but not complete if the rules are the same as Standard C++.		No	
167	14-Jun-04	meeting #5 (WA)	30.3	Technical	L	Brandon Bray	"When the type of a parameter or variable is a type parameter, the declaration of that parameter or variable shall use that type parameter's name without any pointer, reference, or handle declarators." What about cv-qualifiers?		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
168	14-Jun-04	meeting #5 (WA)	30.3	Technical	L	Brandon Bray	Can you take the address of a generic function ins	Meeting #6 (WA): Tentatively decided, NO.	No	
169	14-Jun-04	meeting #5 (WA)	30.3.2	Technical	L	Brandon Bray	The issue raised in 8.15.3 is somewhat answered here. 18.3.6 seems to deal with expanded forms of calls, not expanded forms of function declarations. I interpret the text above as saying that deduction is done as if the function were declared like this: generic <typename ItemType> void PushMultiple(Stack<ItemType>^, ItemType i1, ItemType i2,/* ... */); Is that correct? I think this requires a more detailed description.		No	
170	14-Jun-04	meeting #5 (WA)	30.3.2	Technical	L	Brandon Bray	Something needs to be said about instantiating a generic delegate using a generic function.		No	
171	14-Jun-04	meeting #5 (WA)	30.4.2	Technical	L	Brandon Bray	When are members considered hidden? Is it using the rules described later? Those are described as a		No	
172	14-Jun-04	meeting #5 (WA)	30.4.4	Technical	L	Brandon Bray	Miscellaneous generics issues: 1. I seem to recall discussions of other kinds of constraints (I believe one of them concerned whether you could do a "new T()"). 2. Doesn't there need to be some discussion of how overload resolution works when a function argument has a type parameter as its type? 3. Are the typename and template rules for syntactic disambiguation the same in generics as in templates? Presumably, the lack of specialization would eliminate the need for these. 4. If scope contains a set of overloaded generic functions, is partial ordering used to choose between them? 5. I assume since there is nothing that says otherwise, that generics can be friends of other classes and generics can make other classes, functions, (including generics) friends? 6. If friendship is supported, can a generic first be declared in a friend declaration (suggested answer: no). 7. Standard C++ has restrictions on type parameters such as prohibiting types with no linkage. Does this rule apply to generic arguments? 8. Are there generic conversion functions?		No	
173	14-Jun-04	meeting #5 (WA)	32.1.4	Technical	L	Brandon Bray	To ensure that signatures for the same Type produced by different implementations match, the ordering in such a set of modreqs and modopts is as follows: first modreqs in ascending order by name, then modopts in ascending order by name, with case being significant. [[We need some rule here; is this the one?]].		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
174	14-Jun-04	meeting #5 (WA)	32.1.4	Technical	L	Brandon Bray	If IsBoxed is retained for the standard, we have an ordering issue to consider: Currently, the value-typ		No	
175	14-Jun-04	meeting #5 (WA)	32.1.5.1	Technical	L	Brandon Bray	This modifier [IsBoxed] is a workaround for the MS implementation. Does it have any long-term valu		No	
176	14-Jun-04	meeting #5 (WA)	E	Technical	L	Brandon Bray	Flesh out Future Directions		No	
179	23-Jul-04	TG3 liaison		Technical		Mark Hall	Support for Hide-By-Signature on Methods in ref classes (This would also apply to setter/getter methods for properties.)	See email thread started by Rex J. on Jul 24. Meeting #6 (WA): Some possible ways to address this (and results of a straw poll) are: 1) Support hidebyname only and issue better error messages. [0 in favour] 2) Make all ref class methods be hidebysig; a. Only [0 in favour] b. Default, with an option to select hidebyname [6 in favour] 3) Add hidebysig keyword to allow explicit marking of methods. [0 in favour] with 3 people unsure. We could go two routes: A) Bring hidebysig in via “using” directive to hoist base class/interface names (this is an approximate solution only, as it doesn't allow hoist-by-signature, only hoist-by-name) [0 in favour] B) Do repeated lookup in all base classes (like C#) [8 in favour] Tom circulated the relevant pages from the CLI spec (Partition I, 7.10.4). We need to take into account the CLS rules when resolving this issue. Meeting #7 (WA): Had a brief discussion. No progress.	No	
182	26-Jul-04	phone meeting		Technical	H	Brandon Bray	Discussion of passing a string literal in the presence of overloads taking String^ and const char * (what about char *?)	Meeting #6 (WA): The compiler currently chooses the String^ over the const char*. Involves type deduction across templates and generics. Reassigned from Mark to Brandon. String literal portion of issue 12 was transferred to #182.	No	
183	2-Aug-04	meeting #6 (WA)		Technical	L	Brandon Bray	Overload assignment operator for handles.	Post-meeting #7. MS design team discussed this and believes that we should drop this issue.	No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**

184	2-Aug-04	meeting #6 (WA)		Technical		Herb Sutter	Describe problem with overloading on % vs. &
							
							Herb presented the following code:
							
							#include <iostream>
							using namespace std;
							void f(const int&) { cout << "f(const int&)" << endl; }
							void f(int&) { cout << "f(int&)" << endl; }
							
							void g(int%) { cout << "g(int%)" << endl; }
							void g(int&) { cout << "g(int&)" << endl; }
							
							int main() {
							const int ci = 0;
							int i = 0;
							int^ hi = gcnew int;
							
							f(ci);
							f(i);
							
							g(*hi);
							// g(i); // ambiguous: should g(int&) be preferred?
							}
							
							The following code was his attempt to write an agnostic swap:
							
							template<typename T>
							void swap(T% a, T% b) {
							#if defined NO_PIN_PTR // doesn't work
							T temp = a; a = b; b = temp;
							#elif defined PIN_PTR_BUG // doesn't compile
							T temp = *pin_ptr<T>(a);
							*pin_ptr<T>(*pa) = *pin_ptr<T>(*pb);
									No	

185	2-Aug-04	meeting #6 (WA)		Technical		Herb Sutter	Collapsing reference to reference. (It's in the C++0x spec.)		No	
186	2-Aug-04	meeting #6 (WA)		Technical	L	Brandon Bray	Should we standardize traits?		No	
188	2-Aug-04	meeting #6 (WA)		Technical	H	Brandon Bray	Look at using + to implement String concatenation.		No	
189	2-Aug-04	meeting #6 (WA)		Technical		??	Look at the changes to the grammar for C++0x and note where they affect the C++/CLI grammar.		No	
191	2-Aug-04	meeting #6 (WA)		Technical	M	Brandon Bray	Review the specification checking the usage of accessibility vs. visibility		No	
192	2-Aug-04	meeting #6 (WA)		Technical	L	Brandon Bray	Provide an annex containing the differences between the grammar of Standard C++ and C++/CLI		No	
193	2-Aug-04	meeting #6 (WA)		Technical		Sean Perry	Look at the issue of whether or not the mapping of bool should be implementation-defined.	Meeting 7 (WA): Sean wrote this up and presented it to the full committee on the 2nd day. Based on committee feedback, Sean will revise his paper for future consideration.	No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
194	2-Aug-04	Anthony Williams	15.3.2	Technical		Jonathan Caves	Re Anthony's post to the reflector re "default index	Meeting 7 (WA): Discussed the possibility of disallowing both the default indexed property and operator[].	No	
195	25-Aug-04	Rex Jaeschke	14.1.	Technical	L	Brandon Bray	Separate the list of conversions from the order of preference (such as how Standard C++ separates Standard Conversions from overload resolution).		No	
196	30-Sep-04	meeting #7 (WA)		Technical		Herb Sutter	In native types, % behaves like &.		No	
198	30-Sep-04	meeting #7 (WA)	2	Technical		Tom Plum	Propose wording to require that extensions over and above ISO C++ requirements, be diagnosed.		No	
199	30-Sep-04	meeting #7 (WA)	16.2.1	Technical	R	Brandon Bray	Proof the text on Collection type and how a for each is executed.		No	
200	30-Sep-04	meeting #7 (WA)	19.1	Technical		Herb Sutter	Regarding "Member functions in a native class can be generic", support for this appears to have been added inadvertently. However, is there any user need for it?		No	
201	30-Sep-04	meeting #7 (WA)		Technical					No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
10	4-Dec-03	meeting #1 (TX)	14	Technical	M	Brandon Bray	pull together all the conversion information into one place. Make sure all conversions are covered.		No	
11	4-Dec-03	meeting #1 (TX)	15.3.2	Technical		Steve Adamczyk	comma vs. semicolon as separator in indexed access expressions In indexed access expressions (§15.3.2), comma operators are currently disallowed inside [ ] unless they are enclosed in parentheses. This conflicts with usage in existing template libraries (e.g., Lambda), in which the comma operator occurs inside [ ] without enclosing it in parentheses.	Meeting #2 (HI): Can we treat commas in [ ] not having enclosing parenthesis, in any context, always be treated as punctuators? Yes. Steve will provide words to the editor for this. Meeting #3 (Mel): Steve produced a paper. He reported one outstanding issue: In 15.3.2, "Indexed Access", in the C++/CLI spec is rather vague. There, we have indexed-access: indexed-designator [expression-list] where indexed-access is defined as an additional alternative for postfix-expression: postfix-expression: indexed-access Unfortunately, there isn't any definition of indexed-designator, so I'm not quite sure whether all the multi-dimensional cases are supposed be handled by indexed-designator, leaving the traditional cases to be handled by the original (possibily modified) syntax. An alternative would be not to introduce indexed-access at all, and use the definition postfix-expression: postfix-expression [expression-list] to handle all the cases, for both traditional subscripting and the new C++/CLI indexer references. There was agreement to this, so Steve will update his p	No	
13	4-Dec-03	meeting #1 (TX)	12	Technical	M	Brandon Bray	Add a diagram of the type tree		No	
19	5-Dec-03	meeting #1 (TX)		Technical	L	Brandon Bray	list of overlap between Standard C++ and features proposed by C++/CLI		No	
23	16-Dec-03	Phone meeting	8.2.3	Editorial	H	Brandon Bray	Say more, especially w.r.t the template class array<element-type>.		No	
24	16-Dec-03	Phone meeting	9	Technical	R	Brandon Bray	Review this clause.		No	
25	16-Dec-03	Phone meeting	10	Technical	H	Brandon Bray	Revise this clause by covering topics including application entry point, assembly boundaries, among others.		No	
27	16-Dec-03	Phone meeting	12.13.6	Technical	H	Brandon Bray	Describe how interior_ptr, pin_ptr, array, and safe_cast are template-like with certain constraints.		No	
28	16-Dec-03	Phone meeting	12.3.6	Technical	M	Brandon Bray	Describe how the compiler will need to emit a modopt to distinguish interior_ptr<T> from tracking reference to T (T%) in the metatada.		No	
29	16-Dec-03	Phone meeting	12.3.6.2	Technical	M	Brandon Bray	Spell out target type restrictions (for an interior_ptr)		No	
32	16-Dec-03	Phone meeting	13	Technical		Tom Plum	What, if anything, goes in this clause?		No	
33	16-Dec-03	Phone meeting	14.1.1	Editorial	R	Brandon Bray	Review this subclause.		No	
34	16-Dec-03	Phone meeting	14.4	Editorial	R	Brandon Bray	Review this subclause.		No	
35	16-Dec-03	Phone meeting	15.1	Technical	H	Brandon Bray	The rewrite rules for e[x] (default indexed accesses) are different where there is only one index. This is because there is a potential ambiguity with the C++ operator[]. Is this mentioned elsewhere?		No	
36	16-Dec-03	Phone meeting	15.3.8	Technical	M	Brandon Bray	cv-qualification needs to be considered for dynamic_cast.		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
38	16-Dec-03	Phone meeting	15.3.9	Technical	L	Brandon Bray	Provide a spec for standard typeid (that returns std::type_info) in addition to the new typeid (that returns System::Type).		No	
39	16-Dec-03	Phone meeting	15.3.13	Editorial	H	Brandon Bray	Update this subclause		No	
40	16-Dec-03	Phone meeting	15.4.1.1	Editorial	R	Brandon Bray	Review this subclause.		No	
43	16-Dec-03	Phone meeting	15.11.1	Technical		Mark Hall	Add support for handle equality comparison, and handle ==/!= nullptr, and vice versa.	Meeting #3 (Mel): Had a short discussion. Mark will produce a paper for the May meeting. Meeting #4 (NJ): No progress. To be discussed via email, and at the Jun meeting Meeting #5 (WA): Discussed briefly. Asked Mark to write this up and distribute to the reflector. Phone call Jun 29: This issue was resolved; just needs drafting of final words. Meeting 7 (WA): In the case of if(handle), which conversions are attempted before comparison against nullptr is used? We agreed that if an explicit conversion to bool exists, if(handle) uses that. There is no implicit unboxing. Steve and Mark worked on this and presented it to the full committee on the 2nd day. Based on committee feedback, Mark will write this up for future consideration.	No	
44	16-Dec-03	Phone meeting	15.18	Technical	H	Brandon Bray	Add words to discuss assignment for properties and events from the point of view of the rewrite rules.		No	
47	16-Dec-03	Phone meeting	17	Technical	M	Brandon Bray	Provide text for this clause (Namespaces)		No	
50	16-Dec-03	Phone meeting	18.4	Technical	M	Brandon Bray	Extend declarator-id's by adding a new production that allows default.		No	
52	16-Dec-03	Phone meeting	18.4.2	Technical	H	Brandon Bray	This subclause only covers how the accessor functions must be defined. The expressions clause needs to cover the rewrite rules that call accessor functions.		No	
54	16-Dec-03	Phone meeting	18.5.2	Editorial	R	Brandon Bray	Review this subclause.		No	
55	16-Dec-03	Phone meeting	18.6	Editorial	R	Brandon Bray	Review this subclause.		No	
56	16-Dec-03	Phone meeting	18.7.4	Technical	M	Brandon Bray	Identify when (operator) synthesis would and would not occur.		No	
57	16-Dec-03	Phone meeting	18.6.5.1	Technical	L	Brandon Bray	Writeup op_true and op_false operators		No	
58	16-Dec-03	Phone meeting	18.6.6.1	Technical		Mark Hall	Reword this subclause similarly to the way special member functions are described.	Meeting 7 (WA): ?? To be done in Tue morning work sessions.	No	
59	16-Dec-03	Phone meeting	18.6.6.1	Technical	H	Brandon Bray	Add another subclause to cover the compiler-generated conversion from handle to unspecified bool type.	Meeting 7 (WA): ?? To be done in Tue morning work sessions.	No	
62	16-Dec-03	Phone meeting	18.10.1	Technical	L	Brandon Bray	Add a description that for any value class we have to make the copy before calling member functions.		No	
63	16-Dec-03	Phone meeting	18.11	Technical	H	Brandon Bray	Say more about finalizers (including Dispose/~T and Finalize/!T) and add some examples.		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
65	16-Dec-03	Phone meeting	18.1	Technical		Editor	As a cross-language issue, come up with terminology to distingish between destructors and finalizers. Perhaps "deterministic destructor" vs. "non-deterministic finalizer." Add some text in spec re this, esp. w.r.t C#'s use of destructor.		No	
66	16-Dec-03	Phone meeting	21	Editorial	M	Brandon Bray	Introduce value classes -- Discuss the following: value classes are optimized for small data structures. As such, value classes do not allow inheritance from anything but interface classes. Tie in fundamental classes.		No	
67	16-Dec-03	Phone meeting	21.4.1	Technical	H	Brandon Bray	Add words about instance constructors and static constructor. Value classes cannot have SMFs (specifically, default constructor, copy constructor, assignment operator, destructor, or finalizer. Need to add specification for this along with rationale.		No	
68	16-Dec-03	Phone meeting	22	Technical	L	Brandon Bray	Consider writing some text for this "place-holder" clause. Should this all go in the new annex "Future directions"?		No	
71	16-Dec-03	Phone meeting	23	Editorial	R	Brandon Bray	Will review this whole clause.		No	
74	16-Dec-03	Phone meeting	23.5	Technical	M	Brandon Bray	Write-up array covariance w.r.t arrays.		No	
75	16-Dec-03	Phone meeting	23.6	Technical	M	Brandon Bray	Write up array initialization.		No	
76	16-Dec-03	Phone meeting	24.4	Technical	H	Brandon Bray	Address what happens when a ref class does not implement an interface function (and what happens when a base class has a non-virtual function with the same name).		No	
79	16-Dec-03	Phone meeting	27	Technical	H	Brandon Bray	Cover unification of CLI and Standard C++ exception-handling models, and anything else that might go in this clause. Are exceptions asynchronous now in some cases? Yes they are. (For example, NullReferenceException.)	Meeting #5 (WA): Kevin Free (Microsoft) gave a verbal presentation. catch(...) catches managed and native exceptions. catch(System::Object^) also catches both kinds, but won't invoke the destructor (so can leak). CLI exception handling supports more features than we expose. The issue remained with Brandon to write up, as before.	No	
81	16-Dec-03	Phone meeting	20.5.2	Technical	R	Brandon Bray	Describe MethodImplOption metadata generation.		No	
82	16-Dec-03	Phone meeting	29	Technical	M	Brandon Bray	Flesh out "Templates" clause.		No	
87	16-Dec-03	Phone meeting	A	Technical	L	Brandon Bray	Flesh out "Verifiable code" clause. Describe the dangers of pointer arithmetic and interior_ptrs.		No	
88	16-Dec-03	Phone meeting	B	Technical	L	Editor	Flesh out "Documentation comments" clause.		No	
90	16-Dec-03	Phone meeting	D	Technical		Editor	Add naming guidelines for generics		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
94	29-Jan-04	meeting #2 (HI)		Technical		Mark Hall	Relationship between primitive types and CLI types. The current spec allows the following: int i = 10; String^ s = i.ToString(); Standard C++ doesn't allow member selection on expressions of primitive type. Assuming int maps to System::Int32, just how much alike are these two types? Specifically, when do we treat the primitive as the underlying class.	Meeting 5 (WA): Asked Mark to write this up and distribute to the reflector. Please address the side-effect issue; that is, given (i++).ToString, is the increment done? Meeting 7 (WA): ?? To be done in Tue morning work sessions. Re the side-effect, yes, it must be done.	No	
95	29-Jan-04	meeting #2 (HI)	10	Technical	H	Brandon Bray	Provide words for #using.		No	
96	29-Jan-04	meeting #2 (HI)	9.1.1	Technical	M	Editor	The spec does not provide a way to use a keyword as an identifier. (VC++ uses the intrinsic __identifier(name) to achieve this; C# uses a leading @.) This is an issue for inter-operability; for example, being a consumer of a public type (written in something other than C++) that has a name (or contains a public member that has a name) that is a keyword in C++.	Meeting #8 (WA): It was proposed we support the intrinsic approach, accepting __intrinsic(x), where x is a string literal or an identifier. String version is reserved for implementers.	No	
97	29-Jan-04	meeting #2 (HI)		Technical		Editor	Overloading on arity. (This is a liaison issue with TG3.) The issue involves the overloading of a non-generic type with a one or more generic types of the same name in the same namespace. For example, the following is permitted by the CLS: ref class X { /*…*/ }; generic<typename T> /*…*/ ref class X { /*…*/ }; generic<typename T, typename U> /*…*/ ref class X { /*…*/ };	Meeting 3 (Mel): Herb presented this issue, which was then reassigned to Brandon. Meeting 5 (WA): In this version, we'll support a generic and non-generic version of a type in the same namespace, but not in different namespaces. There was a discussion about using something like "using generic x::y" to provide cross-namespace support as well. Rex to work with Brandon to get this into the draft. Meeting 7 (WA): Herb reported that the MS implementation can consume same-named generics that overload on arity in the same assembly, but it cannot create them.	No	
98	29-Jan-04	meeting #2 (HI)	30	Technical	R	Brandon Bray	Restrictions on generics re generic code generation. The current generics clause needs to be fleshed out, especially w.r.t how overload resolution works within the CLI.	Meeting #2 (HI): Brandon will write a paper on this. Meeting #4 (NJ): The fleshing out of Clause 30 is a significant contribution toward this. More work needed in declarations and function calls.	No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
105	29-Jan-04	meeting #2 (HI)	14.5.1	Technical		Mark Hall	Constructors can't be used in casts in managed classes. Should they be allowed in explicit conversions? All managed type constructors being explicit by default. (Already yes, but reconfirm this.)	Meeting #4 (NJ): Steve will send the editor sufficient text to go into the public drop to indicate our intention re this topic. DONE. Meeting 5 (WA): Asked Mark to write this up and distribute to the reflector. Meeting 7 (WA): Steve and Mark worked on this and presented it to the full committee on the 2nd day. Mark will write this up for future consideration.	No	
106	29-Jan-04	meeting #2 (HI)		Technical		Editor	Should >> handled as two tokens rather than one; e.g., List<List<int>>.	Meeting #3 (Mel): Had a short discussion. Tom will produce a paper for the May meeting. Meeting #4 (NJ): TG5 agreed that if a < for a template is seen, and >> that are not inside parentheses, that >> will always be considered to be the closing delimiter of two < symbols, and results in an error if there are not two such corresponding < symbols. Refer to Daveed's paper WG21/N1649 for more information. Meeting #7 (WA): This paper was updated (see N1699). It was discussed in TG5 and will be discussed at the up-coming WG21 meeting, at which TG5 members will participate. Meeting #8 (WA): Daveed presented this at the WG21 meeting this week. He proposed option 1, to which WG21 agreed. He was charged to write the final words.	No	
109	19-Feb-04		12.3.6.3	Technical	L	Brandon Bray	Cover the dangers of pointer arithmetic and interior_ptrs		No	
111	19-Feb-04		15.3.2	Technical	M	Brandon Bray	Need to consider how indexed access expressions are interpreted in templates.		No	
116	19-Feb-04		18.4.2	Technical	H	Brandon Bray	Add some discussion of how accesses to properties are rewritten into accessor functions. This should be covered in rewrite rules in the expressions clause. Note that access checking for whether a property can be written to or read to is done after rewriting and overload resolutions.		No	
117	19-Feb-04		18.4.2	Technical	H	Brandon Bray	The qualified name of a property needs to be described somewhere. Once that happens, how an out-of-class definition is done will already be covered by existing rules.		No	
121	19-Mar-04	meeting #3 (Mel)		Technical		Steve Adamczyk	In the context of Herb's keywords paper (2004-05), Steve will write up the notion "If it can be an identifier, it is."		No	
122	19-Mar-04	meeting #3 (Mel)		Technical		Steve Adamczyk	Write a WG21 paper on extended integer types, promotion rules, costs of conversion, and the like, for the May meeting.	Meeting #4 (NJ): Not yet done, but still planned.	No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**

124	10-Jun-04	Jonathan Caves		Technical		Jonathan Caves	Indexed properties -- Consider the following:
							
							interface class I1 {
							  property int Value;
							};
							
							interface class I2 {
							  property int Value[String^] {
							  int get(String^);
							  void set(String^, int);
							  };
							};
							
							ref class D : I1, I2 {
							  // Implements the properties
							};
							
							D^ d;
							d->Value["Foo"];
							
							The question is what does the last line do?
							
							Which leads to a language design question - what should the complier do when faced with a property followed by a '['
							
							1) Should it look for just parameterized properties and if there isn't one fail - I suspect not
							
							2) Should it look for all properties and if the returned set contains a parameterized property it should prefer it - this sounds like magic to me.
							
							3) Should it look for all properties perform overload resolution across the whole set and it the resulting call is ambiguous then issue an error.
								Meeting #5 (WA): Discussed this. Option #3 preferred.
								
								Meeting 7 (WA): Discussed this in detail.
								
								property int Value[int] {
								  void set(int, int);
								};
								
								x->Value[1] = 4
								is treated as
								x->set_Value(1,4);
								
								--------- 	
								
								property array<int>^ Value {
								  array<int>^ get();
								}
								
								x->Value[1] = 4
								is treated as
								x->get_Value()[1] = 4
								
								--------- 	
								
								property int% Value[int] {
								  int% get(int);
								}
								
								x->Value[1] = 4
								is treated as
								x->get_Value(1) = 4
								
								This construct violates the principle of properties (that of setting/getting the value of some property), so is not to be encouraged; however, it is supported, but no need to consider it further here.
									No	

125	14-Jun-04	meeting #5 (WA)	8.15.3	Technical	M	Brandon Bray	Based on the rules for type deduction in templates, it seems surprising that you can match array<ItemType>^ with an argument of type int. Here is a standard C++ example intended to illustrate the issue:
							 template <class ItemType> struct Stack {};
							 template <class ItemType> struct Array {
							  Array(ItemType);
							 };
							 template <class ItemType>
							 void PushMultiple(Stack<ItemType>, Array<ItemType>);
							 int main() {
							  Stack<int> s;
							  PushMultiple(s, 1); // deduction fails
							  PushMultiple<int>(s, 1);
							 }
							Are the rules for generic different in this area? [There seems to be information related to this in 30.3.2. See that subclause for further comments on this issue.]
									No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
127	14-Jun-04	meeting #5 (WA)	12.3.3	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the modreq IsBoxed shall be emitted (i.e., passing		No	
128	14-Jun-04	meeting #5 (WA)	12.3.6	Technical	L	Brandon Bray	The compiler will need to emit a modopt to distinguish interior_ptr<T> from tracking reference to T (		No	
129	14-Jun-04	meeting #5 (WA)	12.3.7	Technical	L	Brandon Bray	Need to add text to indicate the circumstances under which the modopt IsPinned shall be emitted (i.e.,		No	
130	14-Jun-04	meeting #5 (WA)	14.1.1	Technical	L	Brandon Bray	Separate the list of conversions from the order of preference (such as how Standard C++ separates Sta		No	
131	14-Jun-04	meeting #5 (WA)	15.3.3	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the following type modifiers shall be emitted, and point to each modifier's definition: • IsBoxed i.e., passing a handle to a value type). • IsByValue (i.e., ref class type passed by value). • IsConst (i.e., pointer or reference to a const-qualified type). • IsExplicitlyDereferenced (i.e., interior_ptr as a parameter). • IsImplicitlyDereferenced (i.e., parameter is a reference). • IsLong (i.e., long/unsigned long/long double parameters). • IsExplicitlyDereferenced (i.e., pin_ptr as a parameter). • IsSignUnspecifiedByte (i.e., plain char's sigedness). • IsUdtReturn (i.e., ref class type returned by value). • IsVolatile (i.e., pointer or reference to a volatile-qualified type).		No	
132	14-Jun-04	meeting #5 (WA)	15.3.10	Technical	M	Brandon Bray	Unboxing and boxing are described as preferred user-defined conversions; however, this is incorrect.		No	
133	14-Jun-04	meeting #5 (WA)	15.3.10	Technical	L	Brandon Bray	The null value is converted to the null value of the destination type. This can be unverifiable and migh		No	
134	14-Jun-04	meeting #5 (WA)	16.3.3	Technical	M	Brandon Bray	Need to add text to indicate the circumstances under which the modreq IsUdtReturn shall be emitted (		No	
135	14-Jun-04	meeting #5 (WA)	18	Technical	R	Brandon Bray	This table and corresponding sections should include Special Member Functions (SMFs) like destruct		No	
136	14-Jun-04	meeting #5 (WA)	18.2.1	Technical	L	Brandon Bray	Need to address the following: C++/CLI uses the System::Reflection::DefaultMemberAttribute attribu		No	
138	14-Jun-04	meeting #5 (WA)	18.4	Technical		Mark Hall	Need to write up the restrictions on trivial properties.		No	
139	14-Jun-04	meeting #5 (WA)	18.4	Technical	L	Brandon Bray	We probably should say something about the reserved names get_Item and set_Item, and their relation		No	
142	14-Jun-04	meeting #5 (WA)	18.5.3	Technical	L	Brandon Bray	An event with the new modifier introduces a new event that does not override an event from a base cla		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
143	14-Jun-04	meeting #5 (WA)	18.6	Technical	L	Brandon Bray	The restriction below does not apply to non-static member operators – that need not have a parameter		No	
144	14-Jun-04	meeting #5 (WA)	18.6.1	Technical	L	Brandon Bray	Provide an example for "Homogenizing the candidate overload set".		No	
145	14-Jun-04	meeting #5 (WA)	18.6.5.2	Technical	L	Brandon Bray	Provide C++ names for operator True and False	Meeting #8 (WA): Move to future directions.	No	
148	14-Jun-04	meeting #5 (WA)	20.2	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the following type modifiers shall be emitted, and point to each modifier's definition: • IsConst (i.e., data member involves a cv type). • IsImplicitlyDereferenced (i.e., has a reference type). • IsLong (i.e., long/unsigned long/long double type). • IsSignUnspecifiedByte (i.e., plain char's sigedness). • IsVolatile (i.e., data member involves a cv type).		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
149	14-Jun-04	meeting #5 (WA)	20.3	Technical	L	Brandon Bray	Add text to indicate the circumstances under which the following type modifiers shall be emitted, and point to each modifier's definition: • IsBoxed i.e., passing a handle to a value type). • IsByValue (i.e., ref class type passed by value). • IsConst (i.e., pointer or reference to a const-qualified type). • IsExplicitlyDereferenced (i.e., interior_ptr as a parameter). • IsImplicitlyDereferenced (i.e., parameter is a reference). • IsLong (i.e., long/unsigned long/long double parameters). • IsExplicitlyDereferenced (i.e., pin_ptr as a parameter). • IsSignUnspecifiedByte (i.e., plain char's signedness). • IsUdtReturn (i.e., ref class type returned by value). • IsVolatile (i.e., pointer or reference to a volatile-qualified type).		No	
151	14-Jun-04	meeting #5 (WA)	25.2	Technical	M	Brandon Bray	The note says "pickup the restrictions from page 333 (of Brandon's paperback copy of the C# spec)".		No	
154	14-Jun-04	meeting #5 (WA)	30.1	Technical	R	Brandon Bray	Doesn't the text "a generic name declared in namespace scope or in class scope shall be unique in that		No	
155	14-Jun-04	meeting #5 (WA)	30.1	Technical	R	Brandon Bray	What is a non-generic type? Does it mean that the rules are the same as classes? As template classes?		No	
156	14-Jun-04	meeting #5 (WA)	30.1	Technical		Editor	Can generic types be nested in native classes?		No	
158	14-Jun-04	meeting #5 (WA)	30.1.1	Technical	R	Brandon Bray	The equivalent wording for template parameters in the working paper has been changed to "defines its		No	
159	14-Jun-04	meeting #5 (WA)	30.1.2	Technical	R	Brandon Bray	30.1.2 says "Like templates in Standard C++, within the body of a generic type any usage of the unqualified unadorned name of that type is assumed to refer to the current instantiation." 30.1.3 then goes on to describe "The instance type". Those seem like to different ways of describing the same concept. Can they be unified in some way?		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
160	14-Jun-04	meeting #5 (WA)	30.1.6	Technical	R	Brandon Bray	This subclause describes when a static constructor is invoked. In 18.8, it references the CLI Standard Partition II (10.5.3). Are the rules the same? (Yes) Should this subclause also just reference the CLI spec? There are two sets of behavior; we need to say which one we use.		No	
161	14-Jun-04	meeting #5 (WA)	30.1.7	Technical	M	Brandon Bray	What to say about explicit conversion functions (which can only occur in managed class types)?		No	
162	14-Jun-04	meeting #5 (WA)	30.2.2	Technical	R	Brandon Bray	This subclause lists the types that can and cannot be generic arguments. Fundamental types are not in		No	
163	14-Jun-04	meeting #5 (WA)	30.2.4	Technical	R	Brandon Bray	"The non-inherited members of a constructed type are obtained by substituting, for each generic-parameter in the member declaration, the corresponding generic-argument of the constructed type. The substitution process is based on the semantic meaning of type declarations, and is not simply textual substitution." It would be helpful to explain this in more detail and/or give an example where this makes a difference.		No	
165	14-Jun-04	meeting #5 (WA)	30.3	Technical	L	Brandon Bray	Types not used as a parameter type to a generic function cannot be deduced. Are the nondeduced context rules the same as Standard C++ or not? The sentence before this is true, but not complete if the rules are the same as Standard C++.	Meeting #8 (WA): The intent for V1 is to use the same rules as for templates.	No	
167	14-Jun-04	meeting #5 (WA)	30.3	Technical	L	Brandon Bray	"When the type of a parameter or variable is a type parameter, the declaration of that parameter or variable shall use that type parameter's name without any pointer, reference, or handle declarators." What about cv-qualifiers?		No	

168	14-Jun-04	meeting #5 (WA)	30.3	Technical	L	Brandon Bray	Can you take the address of a generic function ins	Meeting #6 (WA): Tentatively decided, NO.
								
								Meeting #8 (WA): Reconsidered, and now think YES. Consider the following example:
								
								delegate void D(int);
								
								generic <class T>
								void F(T t);
								
								D^ d = gcnew D(&F<int>);
									No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
169	14-Jun-04	meeting #5 (WA)	30.3.2	Technical	L	Brandon Bray	The issue raised in 8.15.3 is somewhat answered here. 18.3.6 seems to deal with expanded forms of calls, not expanded forms of function declarations. I interpret the text above as saying that deduction is done as if the function were declared like this: generic <typename ItemType> void PushMultiple(Stack<ItemType>^, ItemType i1, ItemType i2,/* ... */); Is that correct? I think this requires a more detailed description.		No	
170	14-Jun-04	meeting #5 (WA)	30.3.2	Technical	L	Brandon Bray	Something needs to be said about instantiating a generic delegate using a generic function.		No	
171	14-Jun-04	meeting #5 (WA)	30.4.2	Technical	L	Brandon Bray	When are members considered hidden? Is it using the rules described later? Those are described as a		No	
172	14-Jun-04	meeting #5 (WA)	30.4.4	Technical	H	Brandon Bray	Miscellaneous generics issues: 1. I seem to recall discussions of other kinds of constraints (I believe one of them concerned whether you could do a "new T()"). 2. Doesn't there need to be some discussion of how overload resolution works when a function argument has a type parameter as its type? 3. Are the typename and template rules for syntactic disambiguation the same in generics as in templates? Presumably, the lack of specialization would eliminate the need for these. 4. If scope contains a set of overloaded generic functions, is partial ordering used to choose between them? 5. I assume since there is nothing that says otherwise, that generics can be friends of other classes and generics can make other classes, functions, (including generics) friends? 6. If friendship is supported, can a generic first be declared in a friend declaration (suggested answer: no). 7. Standard C++ has restrictions on type parameters such as prohibiting types with no linkage. Does this rule apply to generic arguments? 8. Are there generic conversion functions?	Meeting #8 (WA): 1. For V1, we can consume and enforce these special constraints, but we can't author them. However, we plan to do so in future, so add this to "Future directions".	No	
173	14-Jun-04	meeting #5 (WA)	32.1.4	Technical	L	Brandon Bray	To ensure that signatures for the same Type produced by different implementations match, the ordering in such a set of modreqs and modopts is as follows: first modreqs in ascending order by name, then modopts in ascending order by name, with case being significant. [[We need some rule here; is this the one?]].		No	
174	14-Jun-04	meeting #5 (WA)	32.1.4	Technical	L	Brandon Bray	If IsBoxed is retained for the standard, we have an ordering issue to consider: Currently, the value-typ		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
175	14-Jun-04	meeting #5 (WA)	32.1.5.1	Technical	L	Brandon Bray	This modifier [IsBoxed] is a workaround for the MS implementation. Does it have any long-term valu		No	
176	14-Jun-04	meeting #5 (WA)	E	Technical	L	Brandon Bray	Flesh out Future Directions		No	
179	23-Jul-04	TG3 liaison		Technical		Mark Hall	Support for Hide-By-Signature on Methods in ref classes (This would also apply to setter/getter methods for properties.)	See email thread started by Rex J. on Jul 24. Meeting #6 (WA): Some possible ways to address this (and results of a straw poll) are: 1) Support hidebyname only and issue better error messages. [0 in favour] 2) Make all ref class methods be hidebysig; a. Only [0 in favour] b. Default, with an option to select hidebyname [6 in favour] 3) Add hidebysig keyword to allow explicit marking of methods. [0 in favour] with 3 people unsure. We could go two routes: A) Bring hidebysig in via “using” directive to hoist base class/interface names (this is an approximate solution only, as it doesn't allow hoist-by-signature, only hoist-by-name) [0 in favour] B) Do repeated lookup in all base classes (like C#) [8 in favour] Tom circulated the relevant pages from the CLI spec (Partition I, 7.10.4). We need to take into account the CLS rules when resolving this issue. Meeting #7 (WA): Had a brief discussion. No progress.	No	
182	26-Jul-04	phone meeting		Technical	H	Brandon Bray	Discussion of passing a string literal in the presence of overloads taking String^ and const char * (what about char *?)	Meeting #6 (WA): The compiler currently chooses the String^ over the const char*. Involves type deduction across templates and generics. Reassigned from Mark to Brandon. String literal portion of issue 12 was transferred to #182.	No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**

184	2-Aug-04	meeting #6 (WA)		Technical		Herb Sutter	Describe problem with overloading on % vs. &
							
							Herb presented the following code:
							
							#include <iostream>
							using namespace std;
							void f(const int&) { cout << "f(const int&)" << endl; }
							void f(int&) { cout << "f(int&)" << endl; }
							
							void g(int%) { cout << "g(int%)" << endl; }
							void g(int&) { cout << "g(int&)" << endl; }
							
							int main() {
							const int ci = 0;
							int i = 0;
							int^ hi = gcnew int;
							
							f(ci);
							f(i);
							
							g(*hi);
							// g(i); // ambiguous: should g(int&) be preferred?
							}
							
							The following code was his attempt to write an agnostic swap:
							
							template<typename T>
							void swap(T% a, T% b) {
							#if defined NO_PIN_PTR // doesn't work
							T temp = a; a = b; b = temp;
							#elif defined PIN_PTR_BUG // doesn't compile
							T temp = *pin_ptr<T>(a);
							*pin_ptr<T>(*pa) = *pin_ptr<T>(*pb);
									No	

185	2-Aug-04	meeting #6 (WA)		Technical		Herb Sutter	Collapsing reference to reference. (It's in the C++0x spec.)		No	
186	2-Aug-04	meeting #6 (WA)		Technical	M	Brandon Bray	Should we standardize traits?		No	
188	2-Aug-04	meeting #6 (WA)		Technical	H	Brandon Bray	Look at using + to implement String concatenation.		No	
189	2-Aug-04	meeting #6 (WA)		Technical		??	Look at the changes to the grammar for C++0x and note where they affect the C++/CLI grammar.		No	
191	2-Aug-04	meeting #6 (WA)		Technical	R	Brandon Bray	Review the specification checking the usage of accessibility vs. visibility		No	
192	2-Aug-04	meeting #6 (WA)		Technical	L	Brandon Bray	Provide an annex containing the differences between the grammar of Standard C++ and C++/CLI		No	
193	2-Aug-04	meeting #6 (WA)		Technical		Sean Perry	Look at the issue of whether or not the mapping of bool should be implementation-defined.	Meeting 7 (WA): Sean wrote this up and presented it to the full committee on the 2nd day. Based on committee feedback, Sean will revise his paper for future consideration.	No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
194	2-Aug-04	Anthony Williams	15.3.2	Technical		Jonathan Caves	Re Anthony's post to the reflector re "default index	Meeting 7 (WA): Discussed the possibility of disallowing both the default indexed property and operator[].	No	
195	25-Aug-04	Rex Jaeschke	14.1.	Technical	L	Brandon Bray	Separate the list of conversions from the order of preference (such as how Standard C++ separates Standard Conversions from overload resolution).		No	
196	30-Sep-04	meeting #7 (WA)		Technical		Herb Sutter	In native types, % behaves like &.		No	
198	30-Sep-04	meeting #7 (WA)	2	Technical		Tom Plum	Propose wording to require that extensions over and above ISO C++ requirements, be diagnosed.		No	
199	30-Sep-04	meeting #7 (WA)	16.2.1	Technical	R	Brandon Bray	Proof the text on Collection type and how a for each is executed.		No	
200	30-Sep-04	meeting #7 (WA)	19.1	Technical		Herb Sutter	Regarding "Member functions in a native class can be generic", support for this appears to have been added inadvertently. However, is there any user need for it?		No	

201	23-Oct-04	meeting #8 (WA)		Technical	H	Brandon Bray	How to accomodate non-CLI calling conventions on other platforms.
							
							Meeting #8 (WA):
							
							delegate void D(int);
							
							generic<class T>
							void F(T t) { System::Console::WriteLine(t-
							>ToString()); }
							
							typedef void (* FP)(int);
							
							void G(FP fp) {
							D^ d = gcnew D(fp);
							d(1010);
							}
							
							int main() {
							D^ d = gcnew D(&F<int>);
							d(42);
							
							FP fp = &F<int>;
							fp(101);
							
							G(&F<int>);
							
							In MS's implementation, need to use __clrcall to indicate the clr calling convention. This lead to a discussion of how to accomodate non-G193CLI calling conventions on other platforms. It was noted that the CLI draft spec, Partition II, 15.3, "Calling convention", states:
							
							"When dealing with methods implemented outside the CLI it is important to be able to specify the calling convention required. For this reason there
									No	

202	23-Oct-04	meeting #8 (WA)		Technical	H	Brandon Bray	Name lookup in managed classes ignores interfaces.		No	

	A	B	C	D	E	F	G	H	I	J
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Priority**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
203	26-Oct-04	Rex Jaeschke	10.1.2	Technical	M	Brandon Bray	[Note: The compiler needs to add typedef members to the class so that template code can use the return type or the parameter types. [[Need more explanation.]] end note]		No	
204	26-Oct-04	Rex Jaeschke	12.2.2	Technical	M	Brandon Bray	Write intro text.		No	
205	26-Oct-04	Rex Jaeschke	15.5	Technical	H	Brandon Bray	15.5 Explicit type conversion (cast notation) The rules in the C++ Standard (§5.4/5) have been extended for C++/CLI by including safe casts before static casts. • a const_cast • a safe_cast • a safe_cast followed by a const_cast • a static_cast • a static_cast followed by a const_cast • a reinterpret_cast • a reinterpret_cast followed by a const_cast [Note: Standard C++ programs remain unchanged by this, as safe casts are ill-formed when either the expression type or target type is a native class. end note] Provide background on the expected behavior and rationale. (Get this from the updated casting proposal.)		No	
206	26-Oct-04	Rex Jaeschke	21.4	Technical	M	Brandon Bray	Simple value classes: Flesh this out.		No	
207	26-Oct-04	Rex Jaeschke	24.2.5	Technical	H	Brandon Bray	Interface member access: Write up.		No	
208	26-Oct-04	Rex Jaeschke	27.2	Technical	L	Brandon Bray	Attribute specification: Write up modules.		No	

Minutes of the: *8th meeting of Ecma TC39-TG5*
held in: *Redmond, WA, USA*
on: *22-23 October, 2004*

Rex Jaeschke
rex@RexJaeschke.com
2004-10-23

1 Opening

Convener Tom Plum welcomed everyone to the eighth meeting of TG5.

1.1 Appointment of Recording Secretary

Rex Jaeschke was appointed.

1.2 Introduction of participants

The participants introduced themselves. Those attending were: Steve Adamczyk (EDG), Jonathan Caves (Microsoft), Mike Cowlishaw (IBM), Francis Glassborow (WG21), Rex Jaeschke (Microsoft), Thorsten Ottosen (WG21), Sean Perry (IBM), P.J. Plauger (Dinkumware), Tana Plauger (Dinkumware), Tom Plum (Plum Hall), Michiel Salters (WG21), and Herb Sutter (Microsoft), Detlef Vollmann (WG21), Christopher Walker (Dinkumware).

1.3 Host facilities/local information

Local information was provided.

2 Adoption of the agenda

Document 2004-41 was approved without objection.

3 Approval of Minutes of previous TG5 meeting

Document 2004-40 was approved without objection.

4 Matters arising from the minutes not covered elsewhere

None.

5 Project Editor's Report – Rex Jaeschke

Rex presented document 2004-43.

Issue 5a: A ref class not marked `abstract` yet having one or more abstract or pure virtual functions, becomes implicitly abstract. In the spirit of encouraging better programming practices, should we require this to be diagnosed, thereby requiring the abstract to be written on the class explicitly? The committee agreed.

Issue 5b: 18.9, "Static constructors", states: "A static constructor can have any *access-specifier*. [*Note:* However, for security reasons, a static constructor should have a `private` *access-specifier*." True, it can have any access-specifier; however, the compiler always emits it in metadata as

`private`. Should we allow it to de declared with any access-specifier other than `private`? The committee agreed that private should be required.

As we reviewed the long long edits, Steve reported that at its meeting earlier this week, WG21 adopted a slightly modified version of that proposal.

Action: Steve will email to Rex the differences in the final WG21 version of long long.

6 Approving tracked changes in latest draft

Document 2003-42 was approved with a number of editorial changes. The following issues were raised:

3: Remove the reference to the C99 standard (see 30.1) below.

8.7 pp 19/line 29: Regarding the constructors for delegates, to "the second is the address of the non-static member function", append "(using the syntax of a pointer to member)".

18.5.3 pp 90: Fix two examples using named overrides on a property rather than on the accessors of that property.

25.1: Prohibit enumerators called __value in native enums as well.

31: Remove all the current text from this clause (and close out Action Items 7 and 84), replacing it with something along the lines of "Apart from what is mentioned in other clauses of this standard, there are no other requirements on a conforming C++/CLI implementation with regard to the Standard C and C++ libraries."

7 Date and place of next meetings

Dec 30, 2004, is the cut-off date for contributions of all non-trivial issues.

Jan 7, 2005, editor will circulate a new draft.

Jan 8-19, 2005, all TG5 members will perform a detailed review

7.1 Next Meeting

January 20-21, 2005: Westfield NJ; hosted by EDG and Dinkumware.

7.2 Future meetings

March 8-9, 2005: Big Island, Hawaii; hosted by Plum Hall
(and, if needed, 1 hour at 9 am on May 11 to review work done post-main meeting)

Vote the spec out of TG5 and then forward to the GA via the TC39 business meeting, to be held the afternoon of March 11.

8 Reports from Liaisons

8.1 TC39 TG3 (CLI) – Rex Jaeschke

Instances of all value types are initializes to all-bits-zero; default constructors are not supported. A recent version of the MS implementation didn't allow T() for T being an arbitrary value type.

It was reported that this has been fixed in the implementation; T() is valid.

Action: Rex will see if the spec needs words for this.

Rex recently distributed the spec for the new type Nullable<T>. It was adopted by TG3 earlier this week. Please send any comments to him.

8.2 SC22/WG21 (C++) – Tom Plum, P.J. Plauger, Tana Plauger, John Spicer, and Steve Adamczyk.

The following issues were discussed at this week's WG21 meeting:

8.2.1 A strong enum type

- Enumerators having scope within parent enum type.
- No conversion to integer.
- Explicit specification of an underlying type.

WG21 Straw vote 9 strongly in favor/6 mildly in favor/3 strongly opposed to using "enum class" as the way to state this new type.

8.2.2 Forwarding constructors

Still under discussion; however, this topic is no longer part of the C++/CLI spec.

8.2.3 nullptr

What is the type of this?

8.2.4 The new form of the for statement

Should TG5 use a syntax for its "for each" that is compatible with that being looked at by WG21. No, TG5 intentionally chose a different syntax to stay out of WG21's way as it resolves this issue.

8.3 TC39 TG2 (C#) – Rex Jaeschke

None.

8.4 ISO/IEC JTC 1/SC 22 – Rex Jaeschke

Rex outlined the Fast-Track schedule, as follows:

1. Mar, 2005: At its semi-annual business meeting, TC39 agrees to forward the final draft based on TG5's recommendation.
2. May, 2005: The Ecma office will notify JTC 1 that Ecma expects to submit the spec via the Fast Track process in Jul, 2005, and provides an advance copy of the draft spec for circulation, as a courtesy.
3. Jun, 2005: At its semi-annual business meeting, the Ecma General Assembly (GA) adopts the submission as an Ecma standard, Version 1, gives it a number, and makes it available for free from the Ecma public website.
4. Jul, 2005: The Ecma standard is submitted to JTC 1 for Fast Track processing. JTC 1 determines that Subcommittee 22 (SC22 — programming languages and environments) is the appropriate home for this, and assigns the task to SC22.
5. Mid-Jul, 2005: SC22 starts a 6-month letter ballot period.
6. While National Bodies (NBs) are reviewing the specs and, ultimately, submitting comments, so too can TG5 via Ecma. TG5 might want to meet in person or have one or more phone conferences to determine what its comments are and its own formal response to those comments.
7. Jan 1, 2006: A JTC 1 ballot resolution meeting date, location, chairman, and project editor are proposed by Ecma.

8. Mid-Jan, 2006: SC22's 6-month letter ballot period ends and all comments are due to JTC 1's ITTF. (All comments must be submitted electronically using a specific Word template.)
9. Feb 1, 2006: JTC 1's ITTF collates all the ballots and their associated comments, and makes them available to the ballot resolution committee (which is, essentially, TG5).
10. Feb 1, 2006, the SC22 Secretariat announces the date and location of the ballot resolution meeting.
11. Feb 1–mid-Mar, 2006: TG5 works on producing formal responses to all public comments.
12. Late Mar, 2006, the ballot resolution meeting is held for x days. Any NB that has voted NO on the ballot must send a representative; otherwise, their NO vote will be ignored. (Assuming that a sufficient number of NBs vote YES initially, or turn their NO to a YES based on decisions made at the ballot resolution meeting, the draft is unofficially an ISO/IEC standard.)
13. Apr, 2006: The project editor integrates all changes based on the ballot resolution meeting, and forwards the revised spec to ITTF for final proofing and processing.
14. May, 2006: the corresponding Ecma standard is revised to match that adopted by ISO/IEC.
15. Late Sep, 2006: the spec is announced as an ISO/IEC standard.
16. Sep, 2006: TC39 votes to forward the revised draft to the Ecma GA for adoption.
17. Sep, 2006: At the annual ISO/IEC JTC 1/SC 22 plenary, I (as Ecma-to-SC22 liaison) request that JTC 1 make available for free, the ISO/IEC version of the standard.
18. Nov, 2006: JTC 1 approves the free availability.
19. Dec, 2006: The Ecma GA adopts Version 2 of the standard, which, except for some typographical and front matter differences, is identical to that from ISO/IEC.

There was a discussion about the possibility of future maintenance of the C++/CLI standard, especially with regard to WG21.

9 Action item and comment spreadsheet review

A walk-through took place with several issues being closed, re-assigned, or re-prioritized. These changes were recorded in the spreadsheet.

96: Using keywords as identifiers: Assigned to the editor to close out.

106: Handling of >> as a single token: Assigned to the editor to close out with new words from Daveed.

10 Any other business

10.1 Distribution of docs to WG21:

Action: Editor will distribute to the TG5 reflector, WD1.8, so members can make it available on their websites for access by WG21 members. Editor will also announce this availability to the liaison email reflector.

Action: Editor will concatenate the PDFs of all docs (except WD1.8) to WG21, and forward to Herb for distribution. (This package will include these draft minutes after TG5 has had a change to review and correct them via email.) This packet will include a document containing URLs from which the latest draft can be obtained.

10.2 Thank meeting host:

Everyone thanked meeting host Microsoft.

11 Adjournment

The meeting was adjourned at 2:45 pm.