Doc No: SC22/WG21/N1675
J16/04-0115
Date: Aug 19, 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 #5

Meeting #6 of Ecma TC39/TG5 (C++/CLI) was held in Redmond, WA, USA, on August 2-3, 2004.

The following TG5 documents are attached to this liaison report:

- TC39-TG5/2004/28 Agenda for the 6th meeting of Ecma TC39 TG5, Redmond, Washington, USA, 2-3 August 2004
- TC39-TG5/2004/29 Minutes of the TG5 phone call of 26 July 2004
- Specification, August 2004
- TC39-TG5/2004/30 **Intentionally omitted (see below)**
- TC39-TG5/2004/31 C++/CLI Specification comments - revision 31 July 2004
- TC39-TG5/2004/32 Minutes of the 6th meeting of TC39-TG5, Redmond, WA, August 2004
- TC39-TG5/2004/33 C++/CLI Specification comments - revision 19 Aug 2004

Document TC39-TG5/2004/30, "Working Draft 1.6 of the C++/CLI Standard, Language" is not included. The primary changes in it involve an overhaul of the grammar. However, the corresponding changes to the narrative to accommodate those grammar changes is not included (that will come in the next draft). As such, it is recommended that you *not* use this draft as the basis for submitting comments relating to that new grammar.

Note that a recent draft of the C++/CLI specification can be found at the following URLs:

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

Agenda for the: 6th meeting of Ecma TC39-TG5

to be held in: Redmond, WA, USA

on: 2-3 August 2004

TIME: **09:00 till 17:00 on Mon 2nd August 2004**
09:00 till 17:00 on Tue 3rd August 2004
[8:30 AM Breakfast, Noon lunch each day]

LOCATION: **Mon 2nd August: Bldg 42, Room 3005**
Tue 3rd August: Bldg 44, Room 1450
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 (2004TG5-027)

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 September 20-21(am only), Redmond, WA; hosted by Microsoft**

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)**
- **8.2.2 Any other WG21 liaison issues**

Minutes of the: *Phone call of Ecma TC39-TG5*
on: *26 July, 2004*

Rex Jaeschke
rex@RexJaeschke.com
2004-07-26

Pacific Time: 10AM - 12PM (Eastern Time: 1PM - 3PM)

Participants:

Those attending were: Steve Adamczyk (EDG), Brandon Bray (Microsoft), Jonathan Caves (Microsoft), Mark Hall (Microsoft), Rex Jaeschke (Microsoft), Sean Perry (IBM), P.J. Plauger (Dinkumware), Tana Plauger (Dinkumware), Tom Plum (Plum Hall), Herb Sutter (Microsoft), and Daveed Vandevoorde (EDG).

Issues

1) Issue #43, "Add support for handle equality comparison, and handle ==/!= nullptr, and vice versa" (Hall)

This was resolved during the Jun 29 phone call.

Action: Mark Hall will write this up for the Aug meeting.

2) Issue #93, "mapping for types" (Plum)

bool:

Can bool map to System::Boolean?

The Mac platform likely uses 32 bits for bool.

Brandon explained the marshaling that currently happens on Windows to handle languages having a representation of boolean that is other than 8 bits.

literals:

Discussion of passing a string literal in the presence of overloads taking String^ and const char *. Which wins? What about String^ and char *?

Brandon prefers keeping current behavior---that of using const char *.

Action: Mark Hall to write this up for the Aug meeting. (new #182)

3) Status of the public drop of WD1.5:

It's available at www.plumhall.com

The package of documents to WG21 contains the cover page of WD1.5 plus a pointer to the plumhall web site.

Other vendors expect to add it to their sites soon.

4) Beta compiler availability:

Expect to have a new distribution available for the Aug meeting.

5) Support for Hide-By-Signature on Methods in ref classes

(See the email thread started by Rex Jaeschke's on Jul 24.)

The intent is that the compiler will allow the generation of implicit using directives, as necessary, to lift base members having the hide-by-sig attribute into the derived class.

Rex raised the issue of wanting to lift only some of the base members.

Mark mentioned a call such as d->F('x'), where d is a Derived^, when Derived has F(int) and Base has F(char). Derived::F(int) is called, not Base::F(char).

In C# lookup, if lookup in the current class fails, compiler looks in the base class, and then its base, ..., until System::Object is searched. However, C++ stops after the current class.

There was a discussion of argument-dependent lookup. Do we want this for ref classes? Very likely Yes!

We need to keep separate the ideas of Writing C++/CLI code that can be consumed by other CLI-based languages, and writing C++/CLI code for other C++/CLI compilers.

Action: Mark Hall to write this up for the Aug meeting. (new issue #179)

Adjournment

Adjourned at 11:10 am

This is a replacement/place-holder for Document TC39-TG5/2004/30, “Working Draft 1.6 of the C++/CLI Standard, Language”, which is **not** included here. The primary changes in it involve an overhaul of the grammar. However, the corresponding changes to the narrative to accommodate those grammar changes is not included (that will come in the next draft). As such, it is recommended that you *not* use this draft as the basis for submitting comments relating to that new grammar.

Note that a recent draft of the C++/CLI specification can be found at the following URLs:

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

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
5	10-Oct-03	Tom Plum		Technical	Tom Plum	While discussing enums (25.1.3) and wchar_t's not being permitted as an underlying type, a discussion arose w.r.t CLI's requiring wchar_t to have the same representation as System::Char; that is, a 16-bit character. This needs further investigation. Possible need to look at/point to the PDTR currently out from WG11 (ISO C). This is part of a more general issue. Do we require exact mapping for types, or do we allow a certain amount of flexibility? See issue #93.	In email on 2003-10-12 Tom Plum wrote: Refining my comments re wchar_t, I see a short-term and a long-term ... Short-term, there's no need to change anything. The 16-bit unicode type is wchar_t in VC++ and in C++/CLI. Long-term, the decision is up to TG5, and depends upon who participates. My own guess is that TG5 in fact will be the first group that has to integrate Unicode 3.1 and 4.0 into its language definition. I suspect that before we're done we'll have four types of character (and literal and C++ string): char - has to be 8 bits to integrate with CLI 'x' "str" string = basic_string<char> wchar_t - implementation's legacy choice of widechar L'x' L"str" wstring = basic_string<wchar_t> char16_t - 16-bit character type, has to be UCS-2 or UTF-16 for CLI u'x' u"str" ustring (?) = basic_string<char16_t> (or string16?) char32_t - 32-bit character type, has to be UTF-32 for CLI U'x' U"str" Ustring (?) = basic_string<char32_t> (or string32?) wchar_t can be the same type as char16_t or char32_t, but isn't required to be	No	
8	4-Dec-03	meeting #1 (TX)	12.1.1	Technical	Steve Adamczyk	64-bit integer mapping. Meeting #1 (Texas): Steve to write a paper for Jan 04 meeting. Done.	Meeting #2 (Hawaii): This paper will be presented at the March meeting of WG21. Let's see how it is received? Meeting #4 (NJ): Steve will suggest how to tighten existing wording w.r.t a 64-bit integer type in the current draft, as part of the cleanup for the public drop. As to how to document the library support has yet to be determined.	No	
10	4-Dec-03	meeting #1 (TX)	14	Technical	Brandon Bray	pull together all the conversion information into one place. Make sure all conversions are covered.		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
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 (Hawaii): 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 (Melbourne): 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 (possibly 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 pr	No	
12	4-Dec-03	meeting #1 (TX)	9	Technical	Tom Plum	Issue of source code/Unicode mapping. What assumptions, if any, should we make about the form of input text? Handling of string literals, character constants, and comments.	Meeting #3 (Melbourne): Had a short discussion. Tom will produce a paper for the May meeting. Meeting #4 (NJ): Tom got more input at this meeting, and will produce a paper for the Jun meeting. DONE (see email "TG5 issue #12 - character sets" from 5/29 EDT) Meeting #5 (Redmond): Discussed Tom's paper in detail. He'll update and recirculate.	No	
13	4-Dec-03	meeting #1 (TX)	12	Technical	Brandon Bray	Add a diagram of the type tree		No	
19	5-Dec-03	meeting #1 (TX)		Technical	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	Brandon Bray	Say more, especially w.r.t the template class array<element-type>.		No	
24	16-Dec-03	Phone meeting	9	Technical	Brandon Bray	Review this clause.		No	
25	16-Dec-03	Phone meeting	10	Technical	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	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	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	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	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
33	16-Dec-03	Phone meeting	14.1.1	Editorial	Brandon Bray	Review this subclause.		No	
34	16-Dec-03	Phone meeting	14.4	Editorial	Brandon Bray	Review this subclause.		No	
35	16-Dec-03	Phone meeting	15.1	Technical	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	Brandon Bray	cv-qualification needs to be considered.		No	
38	16-Dec-03	Phone meeting	15.3.9	Technical	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	Brandon Bray	Update this subclause		No	
40	16-Dec-03	Phone meeting	15.4.1.1	Editorial	Brandon Bray	Review this subclause.		No	
42	16-Dec-03	Phone meeting	15.4.6	Technical	Brandon Bray	Define the grammar for gcnew array, and describe array creation expression.		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 (Melbourne): 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 (Redmond): 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.	No	
44	16-Dec-03	Phone meeting	15.18	Technical	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	Brandon Bray	Provide text for this clause		No	
48	16-Dec-03	Phone meeting	18.3.1	Technical	Editor	Explain the difference between using 'override' and '= function-name'; one creates an .override directive in CIL, the other does not.		No	
50	16-Dec-03	Phone meeting	18.4	Technical	Brandon Bray	Extend declarator-id's by adding a new production that allows default.		No	
51	16-Dec-03	Phone meeting	18.4	Technical	Brandon Bray	The grammar for indexer-parameter-declaration does not allow handles or pointers, but full declarators are not needed. The grammar should allow a simpler sequence of ptr-operator.		No	
52	16-Dec-03	Phone meeting	18.4.2	Technical	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	Brandon Bray	Review this subclause.		No	
55	16-Dec-03	Phone meeting	18.6	Editorial	Brandon Bray	Review this subclause.		No	
56	16-Dec-03	Phone meeting	18.6.4	Technical	Brandon Bray	Identify when synthesis would and would not occur.		No	
57	16-Dec-03	Phone meeting	18.6.5.1	Technical	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.		No	
59	16-Dec-03	Phone meeting	18.6.6.1	Technical	Mark Hall	Add another subclause to cover the compiler-generated conversion from handle to unspecified bool type.		No	
60	16-Dec-03	Phone meeting	18.9	Technical	Brandon Bray	Add grammar for literal-constant-initializer = Standard C++ constant-initializer + float/double + String + nullptr.		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
62	16-Dec-03	Phone meeting	18.10.1	Technical	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	Brandon Bray	Say more about finalizers (including Dispose/~T and Finalize/!T) and add some examples.		No	
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	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	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	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	Brandon Bray	Will review this whole clause.		No	
74	16-Dec-03	Phone meeting	23.5	Technical	Brandon Bray	Look at array covariance w.r.t arrays having copy constructors.		No	
75	16-Dec-03	Phone meeting	23.6	Technical	Brandon Bray	Write up array initialization.		No	
76	16-Dec-03	Phone meeting	24.4	Technical	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	
78	16-Dec-03	Phone meeting	26.1	Technical	Brandon Bray	Redo the grammar for delegate-definition, and find a place for it in the type tree. Replace all uses of "return-type" with appropriate production.		No	
79	16-Dec-03	Phone meeting	27	Technical	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 (Redmond): 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	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
81	16-Dec-03	Phone meeting	20.5.2	Technical	Brandon Bray	Describe MethodImplOption metadata generation.		No	
82	16-Dec-03	Phone meeting	29	Technical	Brandon Bray	Flesh out "Templates" clause.		No	
87	16-Dec-03	Phone meeting	A	Technical	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	Brandon Bray	Flesh out "Documentation comments" clause.		No	
90	16-Dec-03	Phone meeting	D	Technical	Editor	Add naming guidelines for generics		No	

92	29-Jan-04	meeting #2 (HI)		Technical	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 {
						    Size get() { return s_; }
								No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
93	29-Jan-04	meeting #2 (HI)	12.1	Technical	Tom Plum	Do we require exact mapping for types, or do we allow a certain amount of flexibility? Should the size and representation of types long, long long, and long double (as well as wchar_t, see issue #5) be implementation-defined. Should all (or almost all) of the fundamental types being implementation-defined. The CLI types System::Single and System::Double require IEEE (IEC 559) representation. On many systems these naturally map to float and double, respectively. However, the IBM 390 does not used IEEE format for either of these types. A C++/CLI program running in that environment would want float/double to map to 390 types, so there would need to be a conversion to/from the CLI floating types. In order to encourage the writing of portable code, we'd need the largest core of fundamental type mapping as possible; for example, signed and unsigned 8-, 16-, and 32-bit integer mapping.	Meeting #3 (Melbourne): There was a lengthy discussion. No resolution. Meeting #4 (NJ): There was a lengthy discussion. Meeting #5 (WA): There was another lengthy discussion, which resulted in Plum's notes being incorporated into the meeting minutes.	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 (Redmond): 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?	No	
95	29-Jan-04	meeting #2 (HI)	10	Technical	Brandon Bray	Provide words for #using.		No	
96	29-Jan-04	meeting #2 (HI)	9.1.1	Technical	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
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
97	29-Jan-04	meeting #2 (HI)		Technical	Brandon Bray	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 (Melbourne): 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.	No	
98	29-Jan-04	meeting #2 (HI)	30	Technical	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 (Hawaii): Brandon will write a paper on this. Meeting #4 (NJ): The new clause 32 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 (Redmond): Asked Mark to write this up and distribute to the reflector.	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 (Melbourne): 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.	No	
108	19-Feb-04		12.3.6	Technical	Brandon Bray	Provide syntax for interior_ptrs		No	
109	19-Feb-04		12.3.6.3	Technical	Brandon Bray	Cover the dangers of pointer arithmetic and interior_ptrs		No	
110	19-Feb-04		12.3.7.1	Technical	Brandon Bray	Provide syntax for pinning_ptrs		No	
111	19-Feb-04		15.3.2	Technical	Brandon Bray	Need to consider how indexed access expressions are interpreted in templates.		No	
114	19-Feb-04		15.4.6.2	Technical	Brandon Bray	Does new-initializer need to be changed?		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
116	19-Feb-04		18.4.2	Technical	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	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	
118	19-Feb-04		23.1.1	Technical	Editor	Is reference conversion the correct term?	No; it's a handle conversion	No	
119	19-Feb-04		28.5.1.1	Technical	Editor	Check this name (DefaultMember); this attribute might have been renamed in the CLI standard.		No	
120	19-Mar-04	meeting #3 (Mel)		Technical	Tom Plum	Does typename allow us to pursue a containment policy re elaborated specifiers?		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	
123	3-May-04	meeting #4 (NJ)		Technical	Tom Plum	The draft uses the term "constructed type". It was suggested that the corresponding Standard C++ term is"instantiation". Which should we use?		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**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.
						
						Mark Hall says: Jonathan's looking into deferring the
							Meeting #5 (Redmond): Discussed this. Option #3 preferred.	No	

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

125	14-Jun-04	meeting #5 (WA)	8.15.3	Technical	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	

126	14-Jun-04	meeting #5 (WA)	12.1	Technical	Tom Plum	The type long long will be defined by pointing to the paper WG21 N1565. How to do this normatively		No	
127	14-Jun-04	meeting #5 (WA)	12.3.3	Technical	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	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	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	John Spicer	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
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
131	14-Jun-04	meeting #5 (WA)	15.3.3	Technical	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	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	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	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	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	Brandon Bray	Need to address the following: C++/CLI uses the System::Reflection::DefaultMemberAttribute attribu		No	
137	14-Jun-04	meeting #5 (WA)	18.3	Technical	Brandon Bray	Extend the grammar to accommodate attributes on functions.		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	Brandon Bray	We probably should say something about the reserved names get_Item and set_Item, and their relation		No	
140	14-Jun-04	meeting #5 (WA)	18.5	Technical	Brandon Bray	The production event-type has not yet been defined. The syntactic category of this element needs to b		No	
141	14-Jun-04	meeting #5 (WA)	18.5.2	Technical	Brandon Bray	It is a bit strange to define grammar productions for these functions. We probably should either make		No	
142	14-Jun-04	meeting #5 (WA)	18.5.3	Technical	Brandon Bray	An event with the new modifier introduces a new event that does not override an event from a base cl		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
143	14-Jun-04	meeting #5 (WA)	18.6	Technical	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	Brandon Bray	Provide an example for "Homogenizing the candidate overload set".		No	
145	14-Jun-04	meeting #5 (WA)	18.6.5.2	Technical	Brandon Bray	Provide C++ names for operator True and False		No	
146	14-Jun-04	meeting #5 (WA)	18.9	Technical	Brandon Bray	add literal to storage-class-specifier		No	
147	14-Jun-04	meeting #5 (WA)	18.1	Technical	Brandon Bray	add initonly to storage-class-specifier		No	
148	14-Jun-04	meeting #5 (WA)	20.2	Technical	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
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
149	14-Jun-04	meeting #5 (WA)	20.3	Technical	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	
150	14-Jun-04	meeting #5 (WA)	21.4.1	Technical	Brandon Bray	Add words about instance constructors and static constructor.		No	
151	14-Jun-04	meeting #5 (WA)	24.2	Technical	Brandon Bray	The note says "pickup the restrictions from page 333". Brandon, do you have any idea what this page		No	
152	14-Jun-04	meeting #5 (WA)	25.1.3	Technical	Brandon Bray	Complete the production enum-base. Also, since this production is used by both native and CLI enums, yet it's described in the native section, wording might need to be re-arranged to make it read better from both enums' perspectives.		No	
153	14-Jun-04	meeting #5 (WA)	30.1	Technical	Brandon Bray	The text indicates that a generic-declaration may appear in a class scope, but the syntax of member-de		No	
154	14-Jun-04	meeting #5 (WA)	30.1	Technical	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	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	Brandon Bray	Can generic types be nested in native classes?		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
157	14-Jun-04	meeting #5 (WA)	30.1	Technical	Brandon Bray	Type Overloading – This involves overloading on arity, and is currently under investigation. Such a feature permits the following: ref class X {}; generic<typename T> ref class X {}; generic<typename T, typename U> ref class X {};		No	
158	14-Jun-04	meeting #5 (WA)	30.1.1	Technical	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	Brandon Bray	30.1.3 describes "The instance type". These seem like two different ways of describing the same conc		No	
160	14-Jun-04	meeting #5 (WA)	30.1.6	Technical	Brandon Bray	This subclause describes when a static constructor is invoked. In 18.8, it references the CLI Standard		No	
161	14-Jun-04	meeting #5 (WA)	30.1.7	Technical	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	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	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	
164	14-Jun-04	meeting #5 (WA)	30.3	Technical	Brandon Bray	Can a generic function be declared inside a native class? (No) Can generic functions (and member fun		No	
165	14-Jun-04	meeting #5 (WA)	30.3	Technical	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	
166	14-Jun-04	meeting #5 (WA)	30.3	Technical	Brandon Bray	What, if anything, does it mean for a generic function to be static/extern or inline?		No	
167	14-Jun-04	meeting #5 (WA)	30.3	Technical	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	Brandon Bray	Can you take the address of a generic function instance?		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
169	14-Jun-04	meeting #5 (WA)	30.3.2	Technical	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	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	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	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	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
173	14-Jun-04	meeting #5 (WA)	32.1.4	Technical	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	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	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	Brandon Bray	Flesh out Future Directions		No	
177	14-Jun-04	meeting #5 (WA)	E.7	Technical	Brandon Bray	Add text to show the behavior in the CLI (including CIL).		No	
178	14-Jun-04	meeting #5 (WA)	F	Technical	Brandon Bray	Flesh out anything in incompatibilities with Standard C++		No	
179	23-Jul-04	TG3 liaison		Technical	Mark Hall	Support for Hide-By-Signature on Methods in ref classes	See email thread started by Rex J. on Jul 24.	No	
180	14-Jun-04	meeting #5 (WA)	26	Technical	Editor	Committee agreed with Rex's proposal to require that delegates have the optional BeginInvoke and EndInvoke methods for async processing of delegates.	This was reported to TG3 at its Jun 04 meeting, but there were concerns about the Compact Profile's not being required to support these at runtime. Since this is still an open issue in TG3, this issue will remain open in TG5.	No	
181	27-Jun-04			Technical	Tom Plum	Here are Tom's assumptions: C++/CLI will not initially have a built-in type for decimal the way C# has. In C++/CLI, you have to use namespace System::Decimal. The C++/CLI draft doesn't specify anything about semantics of Decimal; the requirements are as given in CLI (TG3). So we benefit from all the work done in TG3 on allowing IEEE Decimal as an alternative to .NET Decimal. Re the methods of the type System::Decimal methods, are they adequate for the C++ programmer, or should the compiler know something special about Decimal?	Phone call Jun 29: discussed Decimal; agreed C++/CLI can just use constructors. Did this address Tom's Q re Decimal's methods?	No	
182	26-Jul-04	phone meeting		Technical	Mark Hall	Discussion of passing a string literal in the presence of overloads taking String^ and const char * (what about char *?)		No	
183									

Minutes of the: *6th meeting of Ecma TC39-TG5*
held in: *Redmond, WA, USA*
on: *2-3 August, 2004*

Rex Jaeschke
rex@RexJaeschke.com
2004-08-03

1 Opening

Convener Tom Plum welcomed everyone to the sixth 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: Brandon Bray (Microsoft), Jonathan Caves (Microsoft), Mark Hall (Microsoft), Rex Jaeschke (Microsoft), Sean Perry (IBM), P.J. Plauger (Dinkumware), Tana Plauger (Dinkumware), Tom Plum (Plum Hall), John Spicer (EDG), Herb Sutter (Microsoft), and Anson Tsao (Microsoft).

1.3 Host facilities/local information

Local information was provided.

2 Adoption of the agenda

Document 2004-28 was approved without objection; several issues (9.10, 9.11, 9.12, 9.13, 9.14, 9.15, and 9.16) were added.

3 Approval of Minutes of previous TG5 meeting

Document 2004-27 was approved without objection.

4 Matters arising from the minutes not covered elsewhere

None.

5 Project Editor's Report – Rex Jaeschke

Rex gave a verbal report (there is no document version, as the only work done with the new WD was to incorporate the changes approved at the previous meeting and phone calls, and for the overhaul of the grammar).

6 Approving tracked changes in latest draft

Document 2003-30. The document was approved with a number of editorial changes. The review raised the following issues:

9.1.1 The definition of *elaborated-type-specifier* has changed post 2002 C++ std. Should we use the new definition?

We'll have an annex that identifies the things we're tracking in C++0x that differ from Std C++.

Action: Editor to add an annex identifying behavior that is implementation-defined, undefined, or unspecified.

Action: Brandon will review the specification checking the usage of accessibility vs. visibility.

Action: Brandon will provide an annex containing the differences between the grammar of Standard C++ and C++/CLI.

Action: Sean to look at the issue of whether or not the mapping of bool should be implementation-defined.

7 Date and place of next meetings

7.1 Next Meeting

September, 2004. Redmond, WA; hosted by Microsoft.

9/20, Mon: TG5 (C++/CLI)
9/21, Tue: TG5 (C++/CLI) -- concurrent with TG2
9/21, Tue: TG2 (C#) -- concurrent with TG5
9/22, Wed: TG2 (C#)
9/23, Thu: TG3 (CLI)
9/24, Fri morning: TG3 (CLI)
9/24, Fri afternoon: TC39 business meeting

7.2 Future meetings

October, 2004. Redmond, WA; hosted by Microsoft.

10/22, Fri pm: TG5 (immediately following WG21)
10/23, Sat, all day: TG5

Action: Herb to arrange the meeting facility.

(tentative meeting) **November/December, 2004** NJ

(tentative meeting) **January/February, 2004** Big Island, Hawaii (tentative)

March, 2005
Vote spec out of TG5 and then forward to the GA via the TC39 business meeting.

8 Reports from Liaisons

8.1 TC39 TG3 (CLI) – Rex Jaeschke

At the June meeting, Rex asked TG5 to support requiring a conforming compiler to generate the asynch methods BeginInvoke and EndInvoke when compiling a delegate. Although TG5 did this, the issue has been re-opened in TG3 w.r.t to the compact framework.

Action: Rex will track this.

Re the assembly/namespace for the C++-specific modopts and modreqs, we're still waiting on MS's input.

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)

No progress.

8.2.2 Any other WG21 liaison issues

There were none.

9 Action item and comment spreadsheet review

9.1 Restrictions on generics re code gen (#98) – Brandon Bray

No progress.

9.2 Seamless interop (#122) – Adamczyk

No progress.

9.3 wchar_t and other native types (#93) – Tom Plum

Discussed the verifiability of string literals. (This lead to a discussion of #182.)

9.4 Relationship between CLI and primitive types (#94) – Mark Hall

Mark and Steve have been communicating re this. Still pending.

9.5 Taxonomy of types (#13) – Brandon Bray

No progress.

9.6 Unification of exception handling (#79) – Brandon Bray

No progress.

9.7 Program text and Unicode (#12) – Tom Plum

Closed. String literal portion of this issue was transferred to #182.

9.8 Handles, and == (#43) – Mark Hall

No progress.

A new issue was raised (#183): Overload assignment operator for handles. Assigned to Brandon.

A new issue was raised (#187): User-defined assignment operator for handles. Assigned to Brandon.

9.9 Overloading on arity (#97) – Brandon

No progress.

9.10 String literal passed to String^/const char* (#182) – Mark Hall

The compiler currently chooses the String^. This behavior needs to be documented.

Involves type deduction across templates and generics.

Reassigned to Brandon.

9.11 Hide-by-signature (#179) – Mark Hall

TG3 raised this as a liaison issue at its June meeting. Rex started an email thread on this on July 24. C#, VB (and other language) compilers make all methods be HideBySig. However, Standard C++ uses a HideByName approach. Programmers are taking certain C# examples, manually converting them to C++/CLI, with different results. Can/should we “fix” this?

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.

Action: Mark will write this up (based on the result of the straw poll) and will circulate it to the reflector.

There was some discussion of VB’s Shadows keyword.

9.12 Overloading on % vs. & (#184) – Herb Sutter

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);
  *pin_ptr<T>(*pb) = temp;
#else                                 // does compile -- but was it intended?
  pin_ptr<T> pa = &a, pb = &b;
  T temp = *pa; *pa = *pb; *pb = temp;
#endif
}
ref class R { };
class N { };
int main() {
  N n1, n2;
  swap( n1, n2 );
//  swap< int& >( n1, n2 );     // if swap took &'s
  R r1, r2;
  swap( r1, r2 );
//  swap< int% >( r1, r2 );     // if swap took &'s
}
```

Action: Herb to write up this issue.

9.13 Collapsing reference to reference (#185) – Herb Sutter

It's in the C++0x spec.

Action: Herb to write this up.

9.14 Should we standardize traits? (#186) – Brandon Bray

Action: Brandon to write this up.

9.15 String catenation (#188) – Brandon Bray

There was a lengthy discussion on the issue of supporting concatenation of Strings. Although various tokens could be used as concatenation operators, programmers wanted "+".

Action: Brandon to look at using + to implement String concatenation.

9.16 default indexed properties and operator[]

Anthony Williams raised this issue on the email reflector on July 12.

Action: Jon Caves will post a response to the reflector and will provide some replacement words for 15.3.2, especially re the synthesizing of the operator.

9.17 Walk-through of remaining spreadsheet items

A walk-through took place with several issues being closed or re-assigned.

10 Any other business

10.1 Delivery schedule:

The changes made in WD1.6 to the grammar (and to its organization) have far-reaching impact on the surrounding narrative. How will we track and review the changes to this narrative? Rex proposed asking for volunteers to each review one or more different clauses in preparation for an editorial review phone call.

This editorial review phone call will take place on Sep 13 from 10-12 PDT.

Action: Rex will arrange the bridge for this phone call, and will post the phone-in details to the TG5-only reflector.

10.2 Distribution of docs to WG21:

Currently, the narrative of the spec does not match the new grammar. So rather than distribute the current draft (which will undoubtedly confuse readers outside the TG), we agreed to update the public version on various member websites, and point WG21 liaisons to that.

Action: Editor will concatenate the PDFs of all docs (except WD1.6) 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.

10.3 Thank meeting host:

Everyone thanked meeting host Microsoft.

11 Adjournment

The meeting was adjourned at 4:00 pm.

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
8	4-Dec-03	meeting #1 (TX)	12.1.1	Technical	Steve Adamczyk	64-bit integer mapping. Meeting #1 (TX): Steve to write a paper for Jan 04 meeting. Done.	Meeting #2 (HI): This paper will be presented at the March meeting of WG21. Let's see how it is received? Meeting #4 (NJ): Steve will suggest how to tighten existing wording w.r.t a 64-bit integer type in the current draft, as part of the cleanup for the public drop. As to how to document the library support has yet to be determined.	No	
10	4-Dec-03	meeting #1 (TX)	14	Technical	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 pr	No	
13	4-Dec-03	meeting #1 (TX)	12	Technical	Brandon Bray	Add a diagram of the type tree		No	
19	5-Dec-03	meeting #1 (TX)		Technical	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	Brandon Bray	Say more, especially w.r.t the template class array<element-type>.		No	
24	16-Dec-03	Phone meeting	9	Technical	Brandon Bray	Review this clause.		No	
25	16-Dec-03	Phone meeting	10	Technical	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	Brandon Bray	Describe how interior_ptr, pin_ptr, array, and safe_cast are template-like with certain constraints.		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
28	16-Dec-03	Phone meeting	12.3.6	Technical	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	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	Brandon Bray	Review this subclause.		No	
34	16-Dec-03	Phone meeting	14.4	Editorial	Brandon Bray	Review this subclause.		No	
35	16-Dec-03	Phone meeting	15.1	Technical	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	Brandon Bray	cv-qualification needs to be considered.		No	
38	16-Dec-03	Phone meeting	15.3.9	Technical	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	Brandon Bray	Update this subclause		No	
40	16-Dec-03	Phone meeting	15.4.1.1	Editorial	Brandon Bray	Review this subclause.		No	
42	16-Dec-03	Phone meeting	15.4.6	Technical	Brandon Bray	Define the grammar for gcnew array, and describe array creation expression.		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	No	
44	16-Dec-03	Phone meeting	15.18	Technical	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	Brandon Bray	Provide text for this clause		No	
48	16-Dec-03	Phone meeting	18.3.1	Technical	Editor	Explain the difference between using 'override' and '= function-name'; one creates an .override directive in CIL, the other does not.		No	
50	16-Dec-03	Phone meeting	18.4	Technical	Brandon Bray	Extend declarator-id's by adding a new production that allows default.		No	
51	16-Dec-03	Phone meeting	18.4	Technical	Brandon Bray	The grammar for indexer-parameter-declaration does not allow handles or pointers, but full declarators are not needed. The grammar should allow a simpler sequence of ptr-operator.		No	
52	16-Dec-03	Phone meeting	18.4.2	Technical	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	Brandon Bray	Review this subclause.		No	
55	16-Dec-03	Phone meeting	18.6	Editorial	Brandon Bray	Review this subclause.		No	
56	16-Dec-03	Phone meeting	18.6.4	Technical	Brandon Bray	Identify when synthesis would and would not occur.		No	
57	16-Dec-03	Phone meeting	18.6.5.1	Technical	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.		No	
59	16-Dec-03	Phone meeting	18.6.6.1	Technical	Mark Hall	Add another subclause to cover the compiler-generated conversion from handle to unspecified bool type.		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
60	16-Dec-03	Phone meeting	18.9	Technical	Brandon Bray	Add grammar for literal-constant-initializer = Standard C++ constant-initializer + float/double + String + nullptr.		No	
62	16-Dec-03	Phone meeting	18.10.1	Technical	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	Brandon Bray	Say more about finalizers (including Dispose/~T and Finalize/!T) and add some examples.		No	
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	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	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	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	Brandon Bray	Will review this whole clause.		No	
74	16-Dec-03	Phone meeting	23.5	Technical	Brandon Bray	Look at array covariance w.r.t arrays having copy constructors.		No	
75	16-Dec-03	Phone meeting	23.6	Technical	Brandon Bray	Write up array initialization.		No	
76	16-Dec-03	Phone meeting	24.4	Technical	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	
78	16-Dec-03	Phone meeting	26.1	Technical	Brandon Bray	Redo the grammar for delegate-definition, and find a place for it in the type tree. Replace all uses of "return-type" with appropriate production.		No	
79	16-Dec-03	Phone meeting	27	Technical	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	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
81	16-Dec-03	Phone meeting	20.5.2	Technical	Brandon Bray	Describe MethodImplOption metadata generation.		No	
82	16-Dec-03	Phone meeting	29	Technical	Brandon Bray	Flesh out "Templates" clause.		No	
87	16-Dec-03	Phone meeting	A	Technical	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	Brandon Bray	Flesh out "Documentation comments" clause.		No	
90	16-Dec-03	Phone meeting	D	Technical	Editor	Add naming guidelines for generics		No	

92	29-Jan-04	meeting #2 (HI)		Technical	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	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
93	29-Jan-04	meeting #2 (HI)	12.1	Technical	Tom Plum	Do we require exact mapping for types, or do we allow a certain amount of flexibility? Should the size and representation of types long, long long, and long double (as well as wchar_t, see issue #5) be implementation-defined. Should all (or almost all) of the fundamental types being implementation-defined. The CLI types System::Single and System::Double require IEEE (IEC 559) representation. On many systems these naturally map to float and double, respectively. However, the IBM 390 does not used IEEE format for either of these types. A C++/CLI program running in that environment would want float/double to map to 390 types, so there would need to be a conversion to/from the CLI floating types. In order to encourage the writing of portable code, we'd need the largest core of fundamental type mapping as possible; for example, signed and unsigned 8-, 16-, and 32-bit integer mapping.	Meeting #3 (Mel): There was a lengthy discussion. No resolution. Meeting #4 (NJ): There was a lengthy discussion. Meeting #5 (WA): There was another lengthy discussion, which resulted in Plum's notes being incorporated into the meeting minutes. The edits from Plum's subsequent paper were incorporated into WD1.6 for Meeting #6 (WA).	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?	No	
95	29-Jan-04	meeting #2 (HI)	10	Technical	Brandon Bray	Provide words for #using.		No	
96	29-Jan-04	meeting #2 (HI)	9.1.1	Technical	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
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
97	29-Jan-04	meeting #2 (HI)		Technical	Brandon Bray	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.	No	
98	29-Jan-04	meeting #2 (HI)	30	Technical	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.	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.	No	
108	19-Feb-04		12.3.6	Technical	Brandon Bray	Provide syntax for interior_ptrs		No	
109	19-Feb-04		12.3.6.3	Technical	Brandon Bray	Cover the dangers of pointer arithmetic and interior_ptrs		No	
110	19-Feb-04		12.3.7.1	Technical	Brandon Bray	Provide syntax for pinning_ptrs		No	
111	19-Feb-04		15.3.2	Technical	Brandon Bray	Need to consider how indexed access expressions are interpreted in templates.		No	
114	19-Feb-04		15.4.6.2	Technical	Brandon Bray	Does new-initializer need to be changed?		No	
116	19-Feb-04		18.4.2	Technical	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	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
117	19-Feb-04		18.4.2	Technical	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	
118	19-Feb-04		23.1.1	Technical	Editor	Is reference conversion the correct term?	No; it's a handle conversion	No	
119	19-Feb-04		28.5.1.1	Technical	Editor	Check this name (DefaultMember); this attribute might have been renamed in the CLI standard.		No	
120	19-Mar-04	meeting #3 (Mel)		Technical	Tom Plum	Does typename allow us to pursue a containment policy re elaborated specifiers?		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	
123	3-May-04	meeting #4 (NJ)		Technical	Tom Plum	The draft uses the term "constructed type". It was suggested that the corresponding Standard C++ term is"instantiation". Which should we use?		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.	No	

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

125	14-Jun-04	meeting #5 (WA)	8.15.3	Technical	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	

126	14-Jun-04	meeting #5 (WA)	12.1	Technical	Tom Plum	The type long long will be defined by pointing to the paper WG21 N1565. How to do this normatively		No	
127	14-Jun-04	meeting #5 (WA)	12.3.3	Technical	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	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	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	John Spicer	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
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
131	14-Jun-04	meeting #5 (WA)	15.3.3	Technical	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	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	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	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	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	Brandon Bray	Need to address the following: C++/CLI uses the System::Reflection::DefaultMemberAttribute attribu		No	
137	14-Jun-04	meeting #5 (WA)	18.3	Technical	Brandon Bray	Extend the grammar to accommodate attributes on functions.		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	Brandon Bray	We probably should say something about the reserved names get_Item and set_Item, and their relation		No	
140	14-Jun-04	meeting #5 (WA)	18.5	Technical	Brandon Bray	The production event-type has not yet been defined. The syntactic category of this element needs to b		No	
141	14-Jun-04	meeting #5 (WA)	18.5.2	Technical	Brandon Bray	It is a bit strange to define grammar productions for these functions. We probably should either make		No	
142	14-Jun-04	meeting #5 (WA)	18.5.3	Technical	Brandon Bray	An event with the new modifier introduces a new event that does not override an event from a base cl		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
143	14-Jun-04	meeting #5 (WA)	18.6	Technical	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	Brandon Bray	Provide an example for "Homogenizing the candidate overload set".		No	
145	14-Jun-04	meeting #5 (WA)	18.6.5.2	Technical	Brandon Bray	Provide C++ names for operator True and False		No	
146	14-Jun-04	meeting #5 (WA)	18.9	Technical	Brandon Bray	add literal to storage-class-specifier		No	
147	14-Jun-04	meeting #5 (WA)	18.1	Technical	Brandon Bray	add initonly to storage-class-specifier		No	
148	14-Jun-04	meeting #5 (WA)	20.2	Technical	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
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
149	14-Jun-04	meeting #5 (WA)	20.3	Technical	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	
150	14-Jun-04	meeting #5 (WA)	21.4.1	Technical	Brandon Bray	Add words about instance constructors and static constructor.		No	
151	14-Jun-04	meeting #5 (WA)	24.2	Technical	Brandon Bray	The note says "pickup the restrictions from page 333". Brandon, do you have any idea what this page		No	
152	14-Jun-04	meeting #5 (WA)	25.1.3	Technical	Brandon Bray	Complete the production enum-base. Also, since this production is used by both native and CLI enums, yet it's described in the native section, wording might need to be re-arranged to make it read better from both enums' perspectives.		No	
153	14-Jun-04	meeting #5 (WA)	30.1	Technical	Brandon Bray	The text indicates that a generic-declaration may appear in a class scope, but the syntax of member-de		No	
154	14-Jun-04	meeting #5 (WA)	30.1	Technical	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	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	Brandon Bray	Can generic types be nested in native classes?		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
157	14-Jun-04	meeting #5 (WA)	30.1	Technical	Brandon Bray	Type Overloading – This involves overloading on arity, and is currently under investigation. Such a feature permits the following: ref class X {}; generic<typename T> ref class X {}; generic<typename T, typename U> ref class X {};		No	
158	14-Jun-04	meeting #5 (WA)	30.1.1	Technical	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	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	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	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	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	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	
164	14-Jun-04	meeting #5 (WA)	30.3	Technical	Editor	Can a generic function be declared inside a native class? (Yes) Can generic functions (and member fu		No	
165	14-Jun-04	meeting #5 (WA)	30.3	Technical	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	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
166	14-Jun-04	meeting #5 (WA)	30.3	Technical	Editor	What, if anything, does it mean for a generic func	Meeting #6 (WA): all have the usual meaning.	No	
167	14-Jun-04	meeting #5 (WA)	30.3	Technical	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	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	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	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	Brandon Bray	When are members considered hidden? Is it using the rules described later? Those are described as a		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
172	14-Jun-04	meeting #5 (WA)	30.4.4	Technical	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	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	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	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	Brandon Bray	Flesh out Future Directions		No	
178	14-Jun-04	meeting #5 (WA)	F	Technical	Brandon Bray	Flesh out anything in incompatibilities with Standard C++		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
179	23-Jul-04	TG3 liaison		Technical	Mark Hall	Support for Hide-By-Signature on Methods in ref classes	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.	No	
180	14-Jun-04	meeting #5 (WA)	26	Technical	Editor	Committee agreed with Rex's proposal to require that delegates have the optional BeginInvoke and EndInvoke methods for async processing of delegates.	This was reported to TG3 at its Jun 04 meeting, but there were concerns about the Compact Profile's not being required to support these at runtime. Since this is still an open issue in TG3, this issue will remain open in TG5	No	
182	26-Jul-04	phone meeting		Technical	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	Brandon Bray	Overload assignment operator for handles.		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**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);
								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	Brandon Bray	Should we standardize traits?		No	
187	2-Aug-04	meeting #6 (WA)		Technical	Brandon Bray	user-defined assignment operator for handles		No	
188	2-Aug-04	meeting #6 (WA)		Technical	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	
190	2-Aug-04	meeting #6 (WA)		Editorial	Editor	Add an annex identifying behavior that is implementation-defined, undefined, or unspecified.		No	
191	2-Aug-04	meeting #6 (WA)		Technical	Brandon Bray	Review the specification checking the usage of accessibility vs. visibility		No	
192	2-Aug-04	meeting #6 (WA)		Technical	Brandon Bray	Provide an annex containing the differences between the grammar of Standard C++ and C++/CLI		No	

	A	B	C	D	E	F	G	H	I
1	**Date Raised?**	**Issue Raiser?**	**Reference**	**Issue Type**	**Owner**	**Comment**	**Other Remarks**	**Resolved?**	**Postponed?**
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		No	
194	2-Aug-04	Anthony Williams	15.3.2	Technical	Jonathan Caves	Re Anthony's post to the reflector re "default indexed properties" and operator[], will post a response t		No	
195									