

## Re: dynamic\_cast does not work as specified

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2009-01/msg00513.html>

---

- *From:* Joseph M. Newcomer <[newcomer@xxxxxxxxxxxxx](mailto:newcomer@xxxxxxxxxxxxx)>
  - *Date:* Mon, 12 Jan 2009 13:01:26 -0500
- 

See below...

On Mon, 12 Jan 2009 07:42:14 -0600, "Doug Harrison [MVP]" <[dsh@xxxxxxxxx](mailto:dsh@xxxxxxxxx)> wrote:

On Mon, 12 Jan 2009 04:59:06 -0500, Joseph M. Newcomer <[newcomer@xxxxxxxxxxxxx](mailto:newcomer@xxxxxxxxxxxxx)> wrote:

It is not at all clear from the documentation that expression is required to be in the inheritance chain of type-id

Was that a rimshot I heard in the distance? :)

yes, I see now that I can cast to a void\*; I'd read it the other way...)

Consider if I did  
SubThing \* p = (SubThing\*)pHint;  
then p is a pointer to an object (ideally) of type SubThing.

Indeed, pHint had better be of a type that can be legally converted to SubThing\* via a C-style cast.

\*\*\*\*

Indeed, it is; but to avoid the possibility that I'm passing a

```
class SubOtherThing : public Thing
```

by mistake, I wanted to use dynamic\_cast as an additional check on the type. And yes, due to an off-by-1-level computation on the structure, I accidentally sent a SubOtherThing\* to my handler and wasted almost half an hour trying to figure out what I'd done wrong...

\*\*\*\*

## Re: dynamic\_cast does not work as specified

But I wanted the additional typecheck to be done to tell me if the thing that is pointed to is of type (SubThing).

If you want to use dynamic\_cast, but you're coming from void\*, or T\* where T is a lie, as it in your example where MFC makes you use CObject\* as a generic pointer type, you first have to cast the pointer to a known type that has RTTI information attached, i.e. a type that has a vtbl, i.e a type that has a virtual function, the latter being the real requirement. That initial cast cannot be a dynamic cast. Instead, it must be a /valid/reinterpret\_cast. Then you can dynamic\_cast the result of the reinterpret\_cast to another type.

\*\*\*\*\*

To be more specific

```
class Thing {
public:
virtual void Show() PURE;
};

class SubThing : public Thing {
public:
virtual void Show();
};
```

And what you suggested is what I discovered: by casting to a void\* I can then cast it back. (There are actually several virtual methods for class Thing). It was based on the idea that the documentation, which is confusing at best, did not state explicitly that type-id must be a derived class of expression [note that the simple explanation could be done for single-inheritance, then generalized for multiple-inheritance, resulting in a far more readable description].

I'm also trying to figure out how I could write this more clearly for my Errors and Omissions document. Stroustrup 3rd edition wasn't much more help, again trying to define semantics-by-example.

\*\*\*\*\*

What is interesting is that casting to LPVOID makes it work; this points out once again the horror of OnUpdate/UpdateAllViews using a CObject\* instead of an LPVOID.

It is rather presumptuous.

## Re: dynamic\_cast does not work as specified

My suspicion

was that since CObject\* was not part of the hierarchy it was failing, but the documentation does not at all make this clear. The phrase at the end

If type-id is a pointer to an unambiguous accessible direct or indirect base class of

expression, a pointer to the unique subobject of type-id is the result

is not stated as a limitation, but what will happen if typeid is a superclass of expression's type; it does not suggest what happens if typeid is or is not a subclass of expression's type.

You can also perform a cross-cast, e.g. cast between X and Y in this MI hierarchy:

```
#include <stdio.h>
```

```
struct X { virtual void f() {} };
```

```
struct Y {};
```

```
struct Z : X, Y {};
```

```
int main()
```

```
{
```

```
  Z z;
```

```
  X* p = &z;
```

```
  Y* q = dynamic_cast<Y*>(p); // q points to z's Y.
```

```
  X x;
```

```
  X* r = &x;
```

```
  Y* s = dynamic_cast<Y*>(r); // s is set to NULL.
```

```
  printf("%p, %p, %p\n",
```

```
        (void*) static_cast<Y*>(&z),
```

```
        (void*) q,
```

```
        (void*) s);
```

```
}
```

```
X>cl -W4 a.cpp
```

```
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.30729.01 for  
80x86
```

```
X>a
```

```
0012FF30, 0012FF30, 00000000
```

That's why I spoke in terms of the "inheritance graph" instead of base and derived classes.

\*\*\*\*\*

Yes, but I'm not using multiple inheritance. So I wasn't worried about the multiple-inheritance case.

Re: dynamic\_cast does not work as specified

Re: dynamic\_cast does not work as specified

thanks

joe

\*\*\*\*

Joseph M. Newcomer [MVP]

email: newcomer@xxxxxxxxxxxxx

Web: <http://www.flounder.com>

MVP Tips: [http://www.flounder.com/mvp\\_tips.htm](http://www.flounder.com/mvp_tips.htm)

.