How can I confirm in the Windows debugger that I'm looking at a COMDAT-folded function?

July 25, 2025

Some time ago, [we learned about identical COMDAT folding](#) (and also [why it's called "identical COMDAT folding"](#)).

Here's a trick to confirm that you are indeed looking at a COMDAT-folded function: Ask the debugger to show the functions whose names exactly match your suspected function address.

```
0:000> ln 00000001`400b3c50
(00000001`400b3c50)   contoso!std::vector<void *,std::allocator<void *> >::_Tidy
| (00000001`400b3c8c)   contoso!Widget::Toggle
Exact matches:
    contoso!std::vector<void *,std::allocator<void *> >::_Tidy (void)
    contoso!std::vector<_GUID const *,std::allocator<_GUID const *>
>::~vector<_GUID const *,std::allocator<_GUID const *> > (void)
```

These two functions are identical, so the linker merged them. In stack traces, they will show as `contoso!std::vector<void *,std::allocator<void *> >::_Tidy`. It looks like the debugger just picks the first exact match and uses that to represent the group. But if you ask for *all* functions that match the address, then the debugger will cough up the other names for this function.

Note that for this trick to work, you have to use the start of the function, because that's where the exact matches will be.

In general, you don't usually need to go this far. You can infer that you're looking at a COMDAT-folded function because the source code of the calling function is clearly calling some function other than the one shown in the debugger. Even before looking at the source code, you can usually infer it because there would be otherwise no reason for the calling function to be calling the COMDAT-folded function. And the third clue is that the function in question is either small (and is therefore going to match other functions) or is likely to be type-independent (and is therefore going to match other specializations of the same template).

But you can use this trick if you want to be extra sure. (Or if you have to prove it to somebody else.)