Psychic debugging: Why your IContextMenu::InvokeCommand doesn't get called even though you returned success from IContextMenu::QueryContextMenu

devblogs.microsoft.com/oldnewthing/20130201-00

February 1, 2013

Raymond Chen

A customer was having trouble with their `IContextMenu` implementation. They observed that their `IContextMenu::QueryContextMenu` method was being called, but when the user selected their menu item, `IContextMenu::InvokeCommand` was not being called.

Given what you know about shell context menus, you can already direct the investigation. I'll let you read up about it first, especially the part about composition, then we can see how much you've learned.

Welcome back. (Okay, I know you didn't actually do the reading, but I'm welcoming you back anyway.)

Your first theory as to why `IContextMenu::InvokeCommand` is not being called is probably that they returned `S_OK` from `IContextMenu::QueryContextMenu` instead of `MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_NULL, 1)`. That would explain the problem, because a return value of `S_OK` is equivalent to `MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_NULL, 0)`, which means "I successfully added up to zero menu items starting at `idCmdFirst`." When the user picks the menu item they added, the dispatcher will go looking for the corresponding composite menu component, and since they said that they used zero entries, they will naturally never be called, since they disavowed any responsibility for those items.

"Nope, that's not it. We're returning `MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_NULL, 1)`. Any other guesses?"

Oh great, the customer is now playing Twenty Questions.

"I'm going to pose a puzzle with almost no clues, and you get to propose solutions, and I'll say whether or not you're right."

I don't know why customers play this game. Maybe they don't realize that asking for help via email is very different from asking for help face-to-face. In a face-to-face conversation, the answer to a question arrives within seconds, whereas in email it can take hours or days. This means that when the answer finally comes back, the person asking the question has to go back and read the conversation history thread to re-establish context.

Or maybe they think they're going to be disclosing Top Secret Information to Microsoft if they share the code that they can't get to work. Trust me, we don't care about your Top Secret Algorithm for Beating the Stock Market With No Money Down. Go ahead and remove those from the code. We just want to see how you are interfacing with the shell. (And besides, you probably should be removing them anyway, since they are irrelevant and are not part of a minimal program that reproduces the problem.)

What would your next guess be?

"Perhaps you're adding your items with the wrong menu item ID." That would explain the problem, because returning `MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_NULL, 1)` means "I successfully added up to one menu item starting at `idCmdFirst`." But if they didn't actually add it at `idCmdFirst`, then when the user selects the item, it won't be in the range they claimed, and therefore the invoke won't get routed to them.

"Your intuition is wrong again. Here's the code we're using."

I can abide by the "Ha ha, you guessed wrong again!" because it at least prodded them into sharing some code. Not much code, mind you, but at least some.

```
HRESULT SooperSeekrit::QueryContextMenu(
    HMENU hmenu,
    UINT indexMenu,
    UINT idCmdFirst,
    UINT idCmdLast,
    UINT uFlags)
{
  UINT cItemsAdded = 0;
  if (!(uFlags & CMF_DEFAULTONLY) &&
      InsertMenuItem(hmenu,
                     indexMenu,
                     TRUE,
                     &globalMenuItemInfo)) {
    return MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_NULL, 1);
  }
  return MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_NULL, 0);
}
```

The next thing you should have noticed is that they never actually used the `idCmdFirst` parameter. So how could they claim to be adding the items with the correct menu item ID if they ignore the variable that tells them what the correct menu item ID is?

“Could you tell us more about the `globalMenuItemInfo` variable? In particular, what value does it use for the `wID` member, and how do you make sure that it is equal to `idCmd-First`? It seems that you are missing some lines of code here:

```
globalMenuItemInfo.fMask |= MIIM_ID;
globalMenuItemInfo.wID = idCmdFirst;
```

but perhaps there’s something going on that we are missing.”

The customer cheerfully replied, “Oops, sorry, didn’t notice that. Works great now, thanks!”

I didn’t bother to draw their attention to the fact that they lied when they responded to the question “Did you add the menu item with the correct ID?” with “Wrong again! BZZZT!”

The point of today’s story is that you, gentle reader, already know how to debug these types of issues. You just have to take what you know and apply it to the situation at hand. If you know how composite context menu dispatch works, then you can come up with failure modes in which the dispatcher fails to match up the menu item with the component.

Exercise: The customer is still not out of the woods yet. What other bug remains in their `IContextMenu::QueryContextMenu` implementation?

Raymond Chen

Follow