The PSN_SETACTIVE notification is sent each time your wizard page is activated

devblogs.microsoft.com/oldnewthing/20111021-00

October 21, 2011

Raymond Chen

A customer had received a number of crashes via Windows Error Reporting and believed that they had found a bug in the tree view common control.

> In our UI, we have a tree view with checkboxes. The tree view displays a fixed item at the top, followed by a variable number of dynamic items. When the user clicks *Next*, we look at the tree view to determine what the user selected. The code goes like this (pseudo):
>
> ```
> htiRoot = GetTreeRoot();
> // First process the fixed item
> htiFixed = GetChild(htiRoot);
> if (IsTreeViewItemChecked(htiFixed)) {
>     .. add the fixed item ...
> }
> // Now process the dynamic items
> hti = GetNextSibling(htiFixed);
> while (hti != NULL) {
>   if (IsTreeViewItemChecked(hti)) {
>     ... add the dynamic item ...
>   }
>   hti = GetNextSibling(hti);
> }
> ```
>
> In the crashes we receive, other variables in the program indicate that there should be only one dynamic item, but our loop iterates multiple times. Furthermore, the first time through the loop, the `hItem` is not the handle to the first dynamic item but is in fact the handle to the fixed item. This naturally results in a crash when we try to treat the fixed item as if it were a dynamic item.
>
> Another thing we noticed is that at the time of the crash, all three variables `htiRoot` `htiFixed`, and `hti` have the same value.
>
> Our attempts to reproduce the problem in-house have been unsuccessful. From our analysis, we believe that the tree view APIs used to obtain handles to children and sibling nodes are misbehaving.

The customer included the crash bucket number, so we were able to connect to the same crash dumps that the customer was looking at.

The first thing to dismiss was the remark that all three of the local variables had the same value. This is to be expected since they have non-overlapping lifetimes, and the compiler decided to alias them all to each other to save memory.

```
...
        lea     eax,[ebp+8]         ; htiRoot
        push    eax
        push    1                   ; some flag
        push    ebx                 ; some parameter
        call    00965fb9            ; GetTreeRoot
        mov     [ebp-2Ch],eax
        test    eax, eax
        jl      00971a49            ; failed - exit
        mov     edi, [_imp__SendMessageW]
        push    4                   ; TVGN_CHILD
        push    110Ah               ; TVM_GETNEXTITEM
        push    dword ptr [ebx+10h] ; window handle
        call    edi                 ; SendMessage
        mov     [ebp+8],eax         ; htiFixed
    ... eliding if (IsTreeViewItemChecked(...)) ...
        jmp     00971a1c            ; enter loop
00971931:
    ... eliding if (IsTreeViewItemChecked(...)) ...
00971a1c:
        push    dword ptr [ebp+8]   ; hti
        push    1                   ; TVGN_NEXT
        push    110Ah               ; TVM_GETNEXTITEM
        push    dword ptr [ebx+10h] ; window handle
        call    edi                 ; SendMessage
        mov     [ebp+8],eax         ; update hti
        test    eax, eax            ; hti == NULL?
        jne     00971931            ; N: continue loop
```

I’ve removed code not directly relevant to the discussion. The point to see here is that the compiler combined all three variables into one physical memory location at `[ebp+8]` since there is no point in the program where more than one of the values is needed at a time. In other words, the compiler rewrote your code like this:

```
hti = GetTreeRoot();
hti = GetChild(hti);
if (IsTreeViewItemChecked(hti)) {
    .. add the fixed item ...
}
while ((hti = GetNextSibling(hti)) != NULL) {
  if (IsTreeViewItemChecked(hti)) {
    ... add the dynamic item ...
  }
}
```

Not only did the compiler merge all your `hti` variables into one, it realized that once it did that, the two calls to `GetNextSibling` could be folded together as well.

Okay, one mystery solved. What about the others?

From studying the crash dump, the shell team determined that the reason the first dynamic item appears to be the fixed item is that the tree view actually has *two* fixed items:

```
003d06d8 Root
+ 003d0a38 "Configuration settings"
+ 003d0888 "Configuration settings"
+ 003d07b0 "Saved game from May 27, 2009 at 2:42 PM (playing as Thor)"
+ 003d0600 "Saved game from May 27, 2009 at 2:42 PM (playing as Thor)"
```

"Configuration settings" is the fixed item, and the saved games are the dynamic items. (This isn't the actual scenario from the customer, but it gets the point across.) The customer was wrong to use the definite article when referring to *the* handle to *the* fixed item, since there are two fixed items here. In a sense, the customer's understanding that there is only one fixed item clouded their ability to debug the problem: When they saw another fixed item, they assumed not that they received another item that was fixed, but rather that they were getting the same fixed item twice.

Seeing that the tree view was being populated twice directed the next step of the investigation: Why?

The code that populates the tree view is called from the wizard page's `PSN_SETACTIVE` notification, and that one piece of information was the last piece of the puzzle.

The `PSN_SETACTIVE` notification is sent each time the wizard or property sheet page is selected as the current page. If the page is activated twice, then you will get two `PSN_SETACTIVE` notifications. The solution was to populate the tree view only the first time the page was activated.

Exercise: What was missing from the customer's testing that prevented them from reproducing the problem in their labs?

Raymond Chen

Follow