You know the answer: Window destruction

devblogs.microsoft.com/oldnewthing/20080102-00

January 2, 2008

Raymond Chen

The following request for assistance came in from a customer, and given what you know about window destruction, you should eventually be able to figure it out yourself once all the pieces are in place, though it takes some time for all the clues to be revealed.

> We are hitting this exception in our program. This is urgent; please give it priority attention.
>
> ```
> 0006f6ac kernel32!InterlockedCompareExchange+0xc
> 0006f6ec comctl32!CImageListBase::IsValid+0x2a
> 0006f6fc comctl32!HIMAGELIST_QueryInterface+0x2c
> 0006f714 comctl32!ImageList_GetBkColor+0x1b
> 0006f724 comctl32!TV_HasTransparentImage+0x1c
> 0006f744 comctl32!TV_SelectItem+0x1a
> 0006f7d4 comctl32!TV_DeleteItemRecurse+0x12a
> 0006f85c comctl32!TV_DeleteItemRecurse+0x58
> 0006f87c comctl32!TV_DeleteItem+0x8c
> 0006f89c comctl32!TV_DestroyTree+0x90
> 0006f900 comctl32!TV_WndProc+0x2e7
> 0006f92c USER32!InternalCallWinProc+0x23
> 0006f9a4 USER32!UserCallWinProcCheckWow+0x14b
> 0006fa00 USER32!DispatchClientMessage+0xda
> 0006fa28 USER32!__fnDWORD+0x24
> 0006fa54 ntdll!KiUserCallbackDispatcher+0x2e
> 0006fa58 USER32!NtUserDestroyWindow+0xc
> 0006faac comctl32!_RealPropertySheet+0x307
> 0006fac0 comctl32!_PropertySheet+0x45
> 0006fad0 comctl32!PropertySheetW+0xf
> 0006fbc4 abc!Wizard::ModalExecute+0x17c
> 0006fc50 abc!RunWizard+0x564
> 0006fcac abc!Start+0x185
> 0006fd70 abc!ABCEntryW+0x2b9
> 0006ff5c abc!wmain+0x7db
> 0006ffa0 abc!__tmainCRTStartup+0x10f
> 0006ffac kernel32!BaseThreadInitThunk+0xe
> 0006ffec ntdll!_RtlUserThreadStart+0x23
> ```

The proximate problem is that the treeview control is trying to use an imagelist that is no longer valid, probably because it has already been destroyed. If you look at the stack, you can see that the treeview control is being destroyed. You might infer this from the function named `TV_DestroyTree`, or if you look at the parameters, which I removed from the stack trace for brevity, you would see that the message is `WM_DESTROY`.

The next step in unwinding the problem is figuring out who destroyed the imagelist while it was still in use. The customer shared their source code, and a little bit of spelunking revealed that it was this function which destroyed the imagelist:

```
void
XYZPage::OnDestroy()
{
   if (m_hImageList)
   {
      ImageList_RemoveAll(m_hImageList);
      ImageList_Destroy(m_hImageList);
   }
}
```

Now all the clues to the puzzle have been laid out on the table. Use your fantastic powers of deduction to see where the customer went wrong. To refresh your memory, you might want to read this old blog entry.

(And now that you've seen and understood this problem, in the future you can jump from the stack trace directly to the conclusion, thereby exhibiting your psychic debugging powers.)

Raymond Chen

Follow