

Re: How to debug load-time DLL exception?

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2005-02/0629.html>

From: Ivan Brugiolo [MSFT] (ivanbrug_at_online.microsoft.com)

Date: 02/12/05

Date: Sat, 12 Feb 2005 12:05:11 -0800

You are forcing a condition that is prohibited by MSDN:

It is safe to call other functions in Kernel32.dll, because this DLL is guaranteed to be loaded in the process address space when the entry-point function is called. It is common for the entry-point function to create synchronization objects such as critical sections and mutexes, and use TLS. Do not call the registry functions, because they are located in Advapi32.dll. If you are dynamically linking with the C run-time library, do not call malloc; instead, call HeapAlloc.

Calling imported functions other than those located in Kernel32.dll may result in problems that are difficult to diagnose. For example, calling User, Shell, and COM functions can cause access violation errors, because some functions in their DLLs call LoadLibrary to load other system components. Conversely, calling those functions during termination can cause access violation errors because the corresponding component may already have been unloaded or uninitialized.

--

This posting is provided "AS IS" with no warranties, and confers no rights. Use of any included script samples are subject to the terms specified at <http://www.microsoft.com/info/copyright.htm>

"Rich S." <RichS@discussions.microsoft.com> wrote in message news:C8FE6634-2384-456C-80BD-3026393567C4@microsoft.com...

> Thank you for informing me that my symbols were not working right! I was
> wondering about that. Anyway, here is the better stack trace:

```
>
> > NTDLL.DLL!_RtlpWaitForCriticalSection@4()
> NTDLL.DLL!_RtlEnterCriticalSection@4()
> ADVAPI32.DLL!_MapPredefinedHandle@8()
> ADVAPI32.DLL!_RegOpenKeyExW@20()
> OLE32.DLL!C2Security::DetermineProtectionMode()
> OLE32.DLL!__initterm()
> OLE32.DLL!__cinit()
> OLE32.DLL!__CRT_INIT@12()
> OLE32.DLL!__DllMainCRTStartup@12()
> NTDLL.DLL!_LdrpCallInitRoutine@16()
> NTDLL.DLL!_LdrpRunInitializeRoutines@4()
> NTDLL.DLL!_LdrpLoadDll@20()
> NTDLL.DLL!_LdrLoadDll@16()
> KERNEL32.DLL!_LoadLibraryExW@12()
> KERNEL32.DLL!_LoadLibraryExA@12()
```

microsoft.public.win32.programmer.kernel: Re: How to debug load-time DLL exception?

```
> KERNEL32.DLL!_LoadLibraryA@4()
> Intercept.dll!CallWndProc()
> Intercept.dll!CallWndProc()
> NTDLL.DLL!_LdrpCallInitRoutine@16()
> NTDLL.DLL!_LdrpRunInitializeRoutines@4()
> NTDLL.DLL!_LdrpLoadDll@20()
> NTDLL.DLL!_LdrLoadDll@16()
> KERNEL32.DLL!_LoadLibraryExW@12()
> KERNEL32.DLL!_LoadLibraryW@4()
> USER32.DLL!_LoadAppDlls@0()
> USER32.DLL!_ClientThreadSetup@0()
> USER32.DLL!___ClientThreadSetup@4()
> NTDLL.DLL!_KiUserCallbackDispatcher@12()
> GDI32.DLL!_GdiDllInitialize@12()
> USER32.DLL!_UserClientDllInitialize@12()
> NTDLL.DLL!_LdrpCallInitRoutine@16()
> NTDLL.DLL!_LdrpRunInitializeRoutines@4()
> NTDLL.DLL!_LdrpInitializeProcess@12()
> NTDLL.DLL!_LdrpInitialize@12()
> NTDLL.DLL!_KiUserApcDispatcher@20()
>
>
> The error seems to center around OLE32.dll. It occurs whenever that
loads.
>
>
> "Pavel Lebedinsky" wrote:
>
> > You need to fix the OS symbols in order to get a stack
> > trace that makes sense. The easiest way to do get the
> > right symbols is by using the symbol server:
> >
> > http://support.microsoft.com/default.aspx?scid=kb;en-us;319037
> >
> > "Rich S." wrote:
> >
> > > Thank you for replying, Pavel.
> > > This is the whole call-stack:
> > >
> > >> NTDLL.DLL!RtlpWaitForCriticalSection()
> > > NTDLL.DLL!NtCreateThread()
> > > OLE32.DLL!CoDisableCallCancellation()
> > > OLE32.DLL!OleSetClipboard()
> > > NTDLL.DLL!RtlUnicodeStringToAnsiString()
> > > NTDLL.DLL!RtlGetDaclSecurityDescriptor()
> > > NTDLL.DLL!RtlGetDaclSecurityDescriptor()
> > > NTDLL.DLL!LdrLoadDll()
> > > KERNEL32.DLL!LoadLibraryW()
> > > USER32.DLL!ClientThreadSetup()
> > > USER32.DLL!GetKeyboardLayout()
> > > USER32.DLL!UserClientDllInitialize()
> > > NTDLL.DLL!RtlUnicodeStringToAnsiString()
> > > NTDLL.DLL!RtlGetDaclSecurityDescriptor()
> > > NTDLL.DLL!CsrClientConnectToServer()
> > > NTDLL.DLL!CsrClientConnectToServer()
> > > NTDLL.DLL!KiUserApcDispatcher()
> > >
> > > it shows only one win32 thread in operation, and it's in this call
stack.
> >
> >
> >
```