(D)COM AMA Overview and Debugging

JeffMill, Mar 2023

COM vs DCOM

COM is local (in-process) – “zero” overhead.

DCOM is remote (out-of-process, remote machine)

General principles of both:

- Location transparency
- Language Neutral (Binary Standard)
- Versioned
- Well-defined Object Model – Discoverable objects (IDL)
- Wire protocol is extensible (TCP, HTTP, etc.)
- Secure by default

Comparison

COM

- Source
- Target
- All in-proc.
- Possibly some COM run-time (e.g. for Automation)

DCOM

- Client
- COM run-time
  - Local object proxy for cross-process
  - Remote object proxy for cross-machine
- Proxy (marshalling)
- DCOM network protocol (RPC, protocol)
- Stub (unmarshalling)
- Local/Remote server object.

COM Apartments

- COM objects live in an “apartment”.
  - Methods can only be called by a thread that belongs to that apartment.
  - Threads outside that apartment must go through proxy.
- Single-Threaded (STA)
  - All objects receive method calls from single thread.
  - Synchronized with Windows message queue
    - *No need to synchronize object access.*
  - Deprecated? Was primarily for Windows UI apps.
- Multi-Threaded (MTA) “Free-threaded”
  - All objects can receive calls from any threads in MTA
  - Objects must handle synchronization.
- Neutral (NTA) Apartment “Rental”
  - MTA-like but can run on any thread type.
  - STA calling MTA would otherwise be marshalled – thread context switch to RPC thread to access MTA objects.

Comexts vs internal structures

My original wukipedia page had instructions to use the “ReservedForOle” field of the TEB, which was specifically for Windows 2000!

```
0:014> dt _TEB 000000d5cee88000 ReservedForOle
svchost!_TEB
   +0x1758 ReservedForOle : 0x00000222`220fbd80 Void
```

Comexts abstracts all of this. Use it.

%SDXROOT%\onecore\com\combase\dcomrem\context.hxx

COM team worked around people reading this value directly by introducing "pCurrentCtxForNefariousReaders" which is placed in that field location.

Correct API is GetCurrentContext()

```
// This function traditionally stored a pointer to the MTA default context in pCurrentCtx if we found it
// to be null (which can happen on implicit MTA threads). As noted in the declaration of SOleTlsData,
// there are nefarious folks who read this value directly, so to ensure compatibility we'll put the value
// where these readers are expecting it. We never read this, since it is not safe to do so on implicit MTA
// threads (could have been set to the MTA default context for a previously valid MTA).
```

WU CSearchCall Breakpoints

bp wuaueng!CSearchCall::Execute

Called during IUpdateSearcher::BeginSearch call.

Called by wuaueng!CClientCallRecorder::BeginFindUpdates

bp wuaueng!CSearchCall::NotifyClient

Called when CSearchCall completes.

Called by CClientCallRecorder::NotifyClient via CSearchCall::Execute

Hands On

MSFT Internal Debugging Tips

Don't use public debugger (winget install 'WinDbg Preview') -- it doesn't include extensions.
Use internal instead:

\\dbg\privates\latest\dbgxcopyinstall.cmd c:\Debuggers_amd64 amd64

\\dbg\privates\latest\dbgxcopyinstall.cmd c:\Debuggers_x86 x86

To use internal symbols:

Connect to MSVPN-Manual

Debug using: **windbg.exe -y 'cache*;SRV*https://symweb' -psn wuauserv**

“cache” will cache symbols in c:/Debuggers/sym, and sources in c:/Debuggers/src

“symweb” is MSFT internal.

Internal Sources

Use the internal debugger! It supports **.srcpath**

Install Research: https://aka.ms/research.selfhost.application

Windows 10 sources (%SDXROOT%: Os.2020 (Git)\vb_release (official)):

WUAPI: %SDXROOT%\onecore\enduser\WindowsUpdate\client\comapi

wuaueng: %SDXROOT%\onecore\enduser\WindowsUpdate\client\engine

COM: %SDXROOT%\onecore\com\combase

Comexts: %SDXROOT%\com\ole32\com\dbgext

References

“wukipedia” page: COM Debugging Tricks

Sample project used: JeffMill/wuclient: Simple WU client app using ATL. (github.com)