開発者必読！

“もしも”のときのデバッグ技法

.NETアプリケーション障害解析

株式会社NTTデータ
飯山 教史
IIYAMA, Takashi

第３回

SOS.DLLを利用した.NETアプリケーションの解析

Level

1 2 3 **4** 5

Technology Tools

- ❑ Visual Basic
- ☑ Visual C#
- ❑ Visual C++
- ❑ SQL Server
- ❑ Oracle
- ❑ Access
- ❑ ASP.NET
- ☑ Other:
  - WinDbg（Debugging Tools for Windows）
  - SOS.DLL

Samples

.NETアプリケーションの解析

前回はWinDbg（Microsoft Debugging Tools for Windows）を利用したアプリケーションの解析について説明した。.NETアプリケーションで発生した例外個所を解析するときにもWinDbgを利用するが、.NETアプリケーションの場合は.NET Frameworkがアプリケーションを管理しているため、これまでと同じコマンドで解析しても同様の結果は得られない。.NETアプリケーションを解析するには「SOS.DLL」と呼ばれるWinDbg拡張モジュールが必要になる。

そこで今回は、SOS.DLLを利用するための環境作りと、SOS.DLLで.NETアプリケーションのクラッシュを解析する方法について説明する。

なぜSOS.DLLが必要なのか？

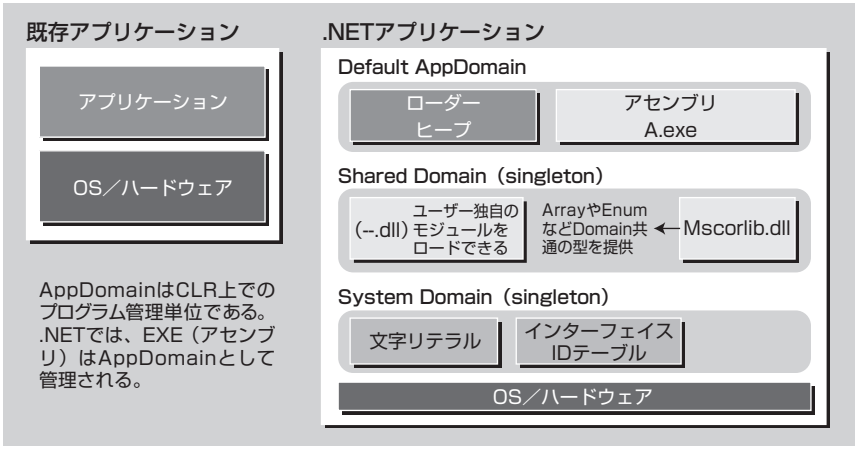
.NETアプリケーションのダンプファイル解析でも、故障部位を特定するだけならこれまでと同様、例外を発生したスレッドを特定してそのスタックをダンプするだけである。同じことをするのになぜSOS.DLLが必要なのだろうか？

それは、.NET FrameworkではCLRがアセンブリを「AppDomain」と呼ばれる単位で管理しているためである。AppDomainとプロセスの関係を示すと図1のようになる。

WinDbgは、Windowsのプロセスベースで管理されているメモリ情報をダンプする機能のみを提供している。そのため、AppDomainの情報をダンプするためにはAppDomain専用のダンプ機能が別途必要になる。たとえば、.NETから新しく導入されたガベージコレクションの状態をダンプする機能などは当然既存のWinDbgでは提供されていないし、スタックやオブジェクトの格納方法も従来とは異なるためWinDbgだけではダンプはできない。

この問題を解決するため、Microsoftは「SOS.DLL」を開発した。SOS.DLL

図1：AppDomainとプロセスの関係



はAppDamainの構造をダンプする高度な機能を持ち、.NETアプリケーションの障害を解析するときには不可欠のツールである[注1]。

SOS.DLLを利用するための準備

SOS.DLLは確かに便利なツールであるが、ひとつ弱点がある。それはWinDbgから利用するためには特殊な方法で起動しなければならないにも関わらず、その起動方法についての情報が書かれているドキュメントが少ない、と言うことである[注2]。ネットで調べた限りでも、WinDbgをダウンロードし、シンボルサーバーを構築した後、どのようにすればSOS.DLLのコマンドを利用できるようになるのかがわからないというエンジニアが多いようだ。

注1）SOSは「Son of Strike」の略で、Strikeという他社製のモジュールのダンプ機能をラップしている。

注2）John RobbinsのBUGSLAYER（MSDN Magazine June 2003）にSOS.DLLの起動方法が書かれている。以下のURLを参照。
http://msdn.microsoft.com/msdnmag/issues/03/06/Bugslayer/default.aspx

▶SOS.DLLをロードする方法

そこで、はじめにWinDbgからSOS.DLLを利用するための環境を構築する方法について説明する。SOS.DLLは.NET Frameworkの一部としてインストールされているので、新たに入手する必要はない。

まず、DOSプロンプトを開き、<Visual Studio .NETのインストールディレクトリ>¥Common7¥Toolsディレクトリ配下の「vsvars32.bat」を実行して必要なディレクトリパスを設定する。

続いて、バッチを実行したDOSプロンプトからWinDbgを起動してCommandウィンドウから、

```
.load sos
```

を実行する。これで、WinDbgでSOS.DLLを利用できるようになる。

この一連の作業を毎回行なうのは面倒なので、私はWinDbgの起動までの作業を行なうバッチファイルを作り、デスクトップにそのバッチファイルのショートカットを用意している。これで、あとは起動したWinDbgで「.load sos」を実行するだけで、SOS.DLLを利用する準備が整う。もちろん「.load sos」を実行しなければ通常のWinDbgとして利用できるので、私はWinDbgを起動するときにはいつもこのバッチを利用している。

▶user.dmpにスタック情報を記録する設定

続いてやらなければならないのは、ダンプファイルにスタック情報を記録する設定である。驚いたことに、デフォルトではuser.dmpに.NETアプリケーションのスタック情報は記録されないのである。

user.dmpにスタック情報を記録するためには、machine.configとレジストリに対して以下のような設定を行なう必要がある。

設定1：machine.config

<SystemRoot>¥Microsoft.NET¥Framework¥v1.1.4322¥CONFIGディレクトリ配下にある「machine.config」を開き、system.windows.formsセクションのjitDebuggingの値を「true」にする（デフォルトはfalse）。

```
<configration>
  ・
  ・
  <system.windows.forms
   jitDebugging="true" />
```

設定2：レジストリ

レジストリエディタを起動し、HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥.NETFrameworkキーのDbgJITDebugLaunchSettingの値を「0」にする。