NOTES:

This information is provided for support purposes only. Each toolset referenced in this document must be used in a manner consistent with its terms of use.

Background

Microsoft Product Support has a large toolset for capturing additional information when trying to determine the root cause of a problem. With respect to our Open Protocols documentation, this document defines three additional methods of data collection that can help us determine the root cause of a problem. These methods are used in addition to providing a network capture of the issue.

These three methods are:

- Use our debugging tools for windows to collect a process memory dump (full or mini)
- Enable ETW tracing events
- Enable iDNA tracing

Below is detailed information on how to use the tools in each scenario and when it is useful to us.

Data collection methods

Use of debugging tools for Windows

Microsoft provides a standard debugger toolset that can be used to collect post-mortem debugging information. This tool is also part of the iDNA toolset which enables the replay of a codepath that leads to a problem or scenario to reproduce. In the context of protocols work, WinDbg is most useful to capture a process memory dump when an exception (first or second chance) occurs. The toolset can be downloaded here.

Once you have installed the toolset on the target host for which you want to capture debugging information, you will then need to find the process ID. You can do so by adding the process ID counter to Task Manager as follows:

Go to ***Start / Run*** and type ***TaskMgr.exe*** to launch Task Manager.

Once in Task Manager, select ***View / Select Columns*** as shown below.

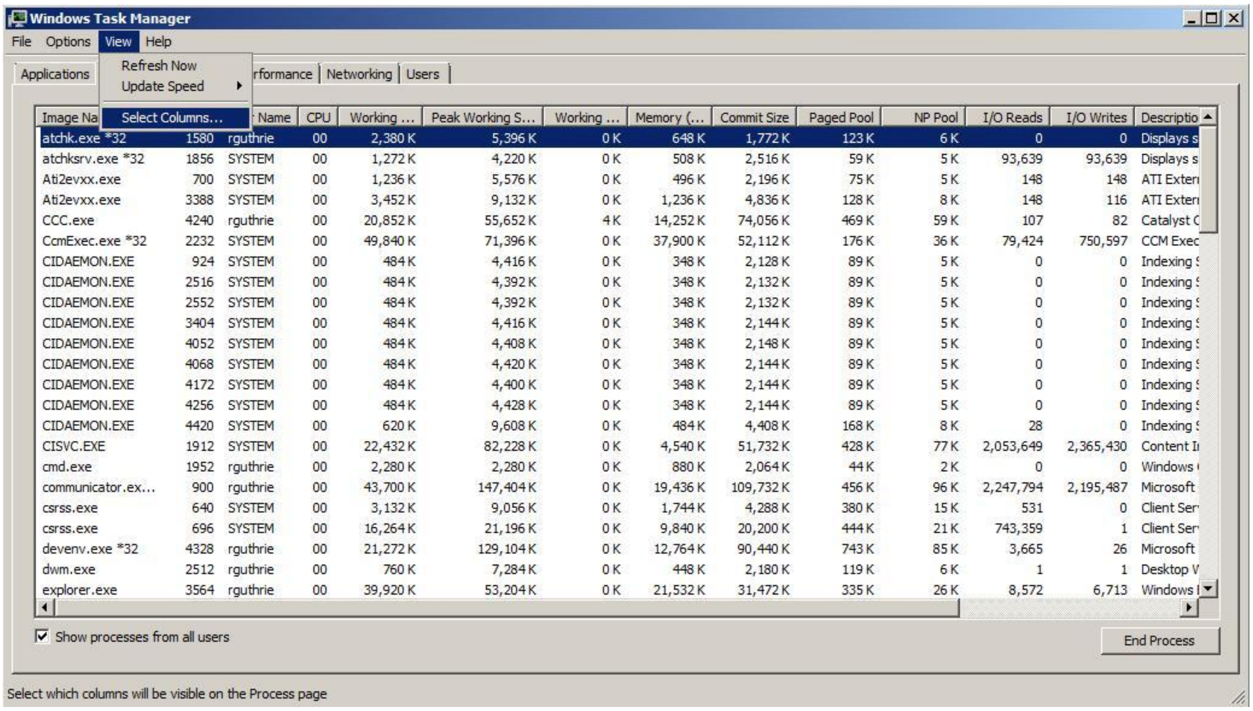
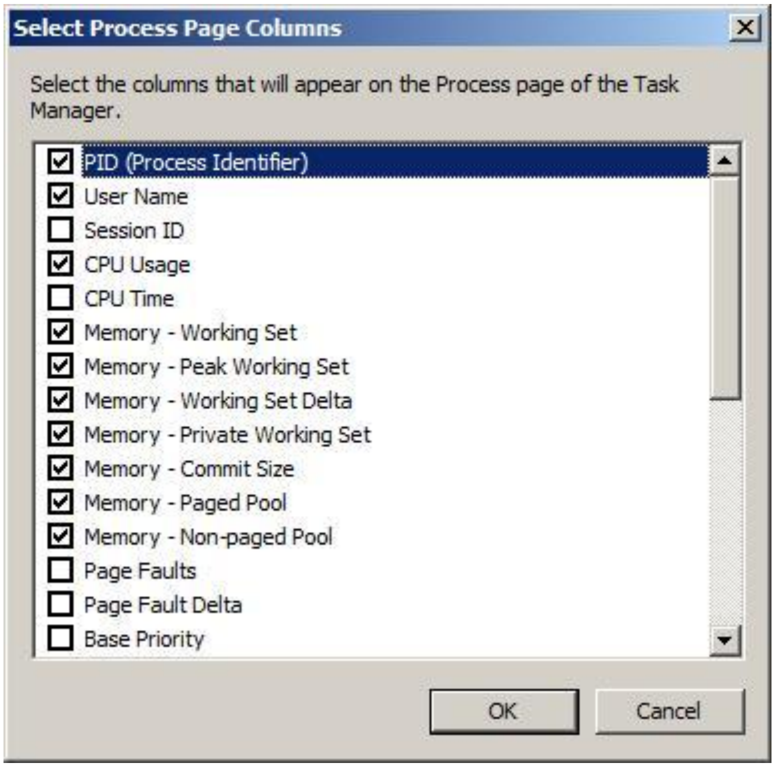


Image Na		r Name	CPU	Working ...	Peak Working S...	Working ...	Memory (...	Commit Size	Paged Pool	NP Pool	I/O Reads	I/O Writes	Descriptio
atchk.exe *32	1580	rguthrie	00	2,380 K	5,396 K	0 K	648 K	1,772 K	123 K	6 K	0	0	Displays s
atchksrv.exe *32	1856	SYSTEM	00	1,272 K	4,220 K	0 K	508 K	2,516 K	59 K	5 K	93,639	93,639	Displays s
Ati2evxx.exe	700	SYSTEM	00	1,236 K	5,576 K	0 K	496 K	2,196 K	75 K	5 K	148	148	ATI Exter
Ati2evxx.exe	3388	SYSTEM	00	3,452 K	9,132 K	0 K	1,236 K	4,836 K	128 K	8 K	148	116	ATI Exter
CCC.exe	4240	rguthrie	00	20,852 K	55,652 K	4 K	14,252 K	74,056 K	469 K	59 K	107	82	Catalyst
CcmExec.exe *32	2232	SYSTEM	00	49,840 K	71,396 K	0 K	37,900 K	52,112 K	176 K	36 K	79,424	750,597	CCM Exec
CIDAEMON.EXE	924	SYSTEM	00	484 K	4,416 K	0 K	348 K	2,128 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	2516	SYSTEM	00	484 K	4,392 K	0 K	348 K	2,132 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	2552	SYSTEM	00	484 K	4,392 K	0 K	348 K	2,132 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	3404	SYSTEM	00	484 K	4,416 K	0 K	348 K	2,144 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	4052	SYSTEM	00	484 K	4,408 K	0 K	348 K	2,148 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	4068	SYSTEM	00	484 K	4,420 K	0 K	348 K	2,144 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	4172	SYSTEM	00	484 K	4,400 K	0 K	348 K	2,144 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	4256	SYSTEM	00	484 K	4,428 K	0 K	348 K	2,144 K	89 K	5 K	0	0	Indexing
CIDAEMON.EXE	4420	SYSTEM	00	620 K	9,608 K	0 K	484 K	4,408 K	168 K	8 K	28	0	Indexing
CISVC.EXE	1912	SYSTEM	00	22,432 K	82,228 K	0 K	4,540 K	51,732 K	428 K	77 K	2,053,649	2,365,430	Content I
cmd.exe	1952	rguthrie	00	2,280 K	2,280 K	0 K	880 K	2,064 K	44 K	2 K	0	0	Windows
communicator.ex...	900	rguthrie	00	43,700 K	147,404 K	0 K	19,436 K	109,732 K	456 K	96 K	2,247,794	2,195,487	Microsoft
csrss.exe	640	SYSTEM	00	3,132 K	9,056 K	0 K	1,744 K	4,288 K	380 K	15 K	531	0	Client Ser
csrss.exe	696	SYSTEM	00	16,264 K	21,196 K	0 K	9,840 K	20,200 K	444 K	21 K	743,359	1	Client Ser
devenv.exe *32	4328	rguthrie	00	21,272 K	129,104 K	0 K	12,764 K	90,440 K	743 K	85 K	3,665	26	Microsoft
dwm.exe	2512	rguthrie	00	760 K	7,284 K	0 K	448 K	2,180 K	119 K	6 K	1	1	Desktop
explorer.exe	3564	rguthrie	00	39,920 K	53,204 K	0 K	21,532 K	31,472 K	335 K	26 K	8,572	6,713	Windows

☑ Show processes from all users

Select which columns will be visible on the Process page

Then select the PID (***Process Identifier)*** column and the select ***Ok***. This is shown below:



Select the columns that will appear on the Process page of the Task Manager.

- ☑ PID (Process Identifier)
- ☑ User Name
- ☐ Session ID
- ☑ CPU Usage
- ☐ CPU Time
- ☑ Memory - Working Set
- ☑ Memory - Peak Working Set
- ☑ Memory - Working Set Delta
- ☑ Memory - Private Working Set
- ☑ Memory - Commit Size
- ☑ Memory - Paged Pool
- ☑ Memory - Non-paged Pool
- ☐ Page Faults
- ☐ Page Fault Delta
- ☐ Base Priority

This will allow you to see the process ID of the process you will capture a crash dump for. To create a crash dump, navigate to the debugging tools directory (selected at install time) and run the command ***adplus –crash –p <INSERT PROCESS ID>***. This will set up the debugger to capture a full memory dump for both first and second chance exceptions. For information on the difference between first and second chance exceptions you can read more here. For more advanced types of debugging scenarios it might also be useful to consider usage of a tool on TechNet called Debug Diag. The details for that tool can be found here.

Typically, we are interested in catching an exception to understand why an interaction between Windows and a system looking to achieve interoperability is failing. For example, a domain trust is established between Windows Active Directory and a third party implementation. An administrator then opens up the Tool Active Directory Users & Computers and tries to browse users in the interoperable implementation. The tool crashes. By using Windbg and its toolset we can capture a process memory dump at the time the exception occurs to try and understand what led to the crash.

Applicable scenarios for interoperability

- Application on Windows Operating System is crashing

Enable iDNA tracing

iDNA Tracing is an extension of Windbg that allows you to monitor and capture activities performed by a process for post mortem recreation of the issue. It records the activity of the process so that it can be replayed offline. This allows the support team to avoid a live debugging scenario in order to reproduce and diagnose the problem. Once it is deemed this is the best course of action, Microsoft will provide you with a zip file containing the iDNA tracing toolset, along with start/stop scripts to capture the particular issue. We will also provide an upload site so that you can send us the information. This will be done on an as-needed basis.

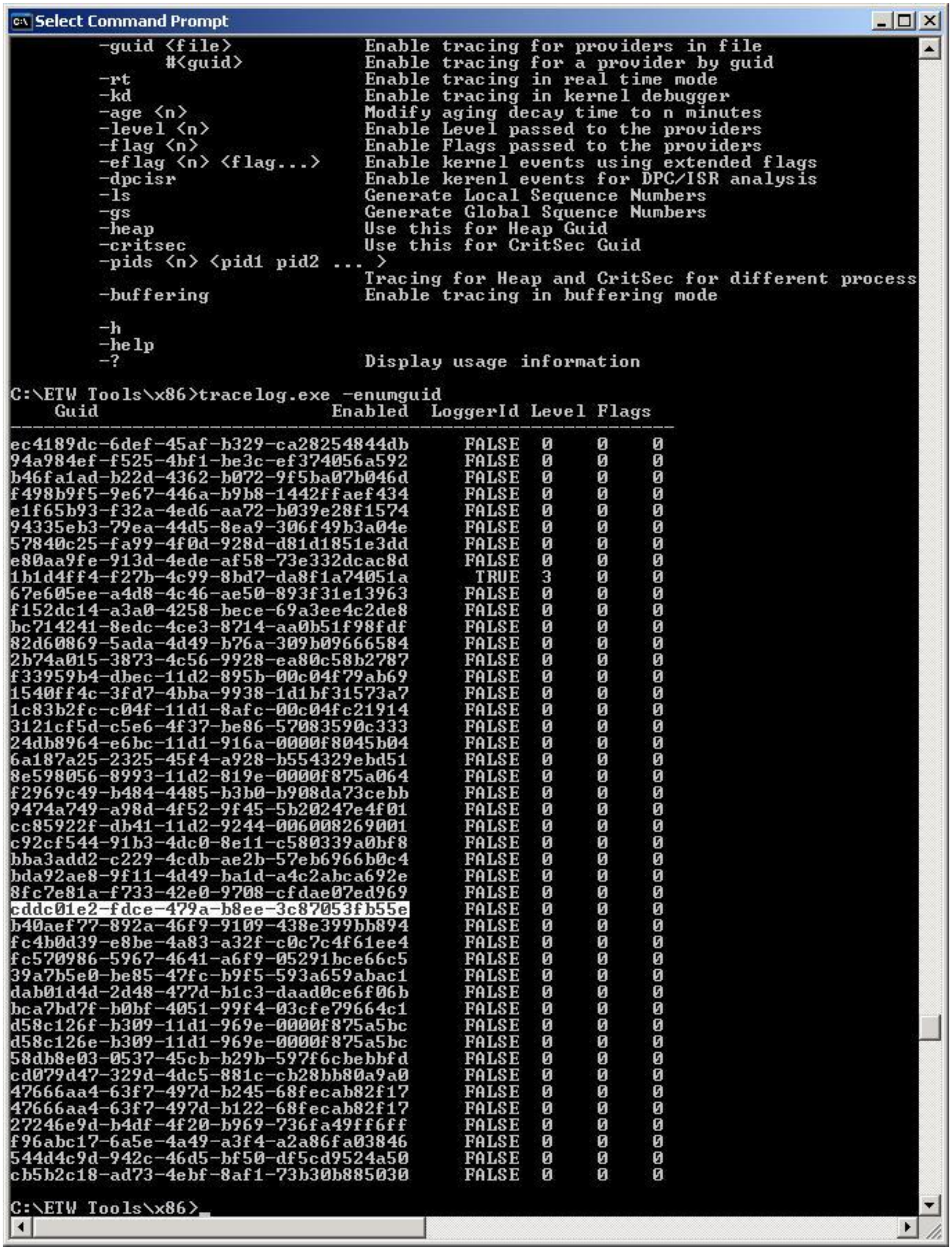
Applicable scenarios for interoperability

- RPC calls failing to produce the expected results
- Server process is failing and crash dump does not lead to resolution of problem

Use of ETW Tracing events

ETW Tracing acts as a method for data collection. The output of data collection is a log file that Microsoft can use to gain more visibility into what the process is doing without the need for a checked or debug build of a .dll. More information on ETW and how it works can be found here on MSDN. ETW tracing requires the user to supply a known GUID(s), along with debugging level (more information can be found here on debugging levels) and several other parameters such as output directory. The operating system will then begin collecting trace information related to that logging level. The resulting .etl file can then be analyzed by Microsoft post-mortem to the problem that is being debugged.

ETW Tracing provides a means of collecting data using a standard toolset that ships with Windows. Using this toolset only to collect diagnostic data would not be considered reverse engineering. Tracelog.exe is a “collection only” tool and is not used in the analysis of collected data. Logman.exe is a tool that can be executed from the command line in Windows to determine the available ETW providers and their associated GUIDs. More information on Logman can be found here. When appropriate, a Microsoft support engineer will provide the command line parameters to enable trace logging for you to execute. In the tracelog example below the highlighted GUID corresponds to RDBSS.



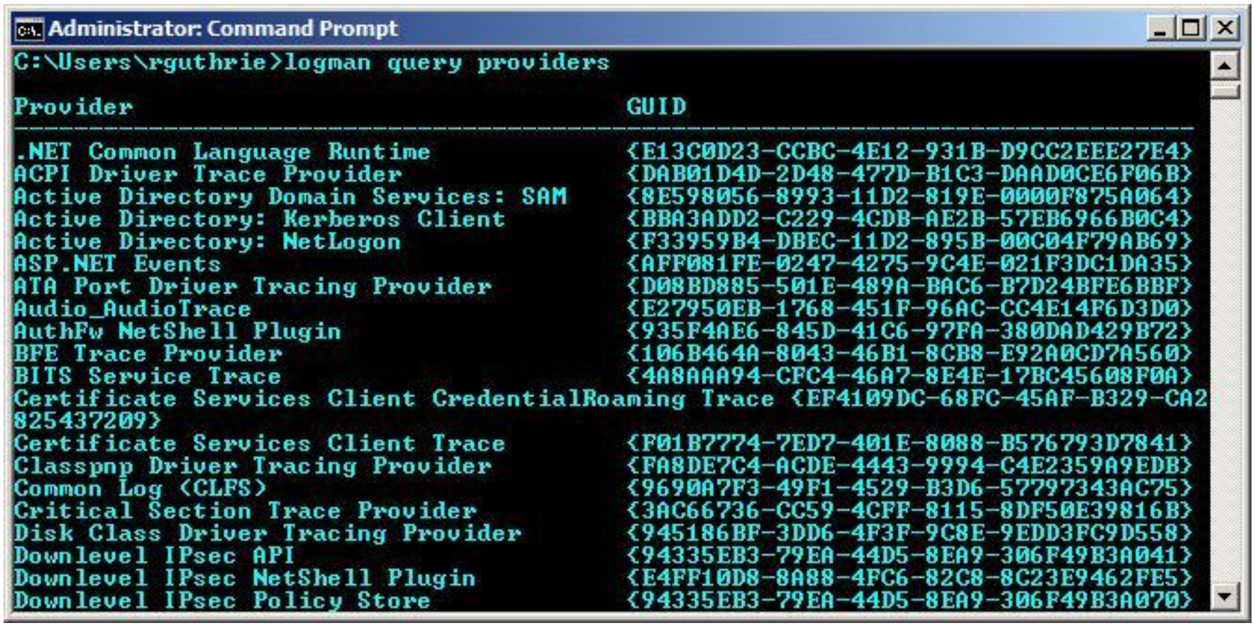
```
        -guid <file>            Enable tracing for providers in file
              #<guid>           Enable tracing for a provider by guid
        -rt                     Enable tracing in real time mode
        -kd                     Enable tracing in kernel debugger
        -age <n>                Modify aging decay time to n minutes
        -level <n>              Enable Level passed to the providers
        -flag <n>               Enable Flags passed to the providers
        -eflag <n> <flag...>    Enable kernel events using extended flags
        -dpcisr                 Enable kerenl events for DPC/ISR analysis
        -ls                     Generate Local Sequence Numbers
        -gs                     Generate Global Squence Numbers
        -heap                   Use this for Heap Guid
        -critsec                Use this for CritSec Guid
        -pids <n> <pid1 pid2 ... >
                                Tracing for Heap and CritSec for different process
        -buffering              Enable tracing in buffering mode

        -h
        -help
        -?                      Display usage information

C:\ETW Tools\x86>tracelog.exe -enumguid
    Guid                     Enabled  LoggerId Level Flags
------------------------------------------------------------
ec4189dc-6def-45af-b329-ca28254844db     FALSE  0    0    0
94a984ef-f525-4bf1-be3c-ef374056a592     FALSE  0    0    0
b46fa1ad-b22d-4362-b072-9f5ba07b046d     FALSE  0    0    0
f498b9f5-9e67-446a-b9b8-1442ffaef434     FALSE  0    0    0
e1f65b93-f32a-4ed6-aa72-b039e28f1574     FALSE  0    0    0
94335eb3-79ea-44d5-8ea9-306f49b3a04e     FALSE  0    0    0
57840c25-fa99-4f0d-928d-d81d1851e3dd     FALSE  0    0    0
e80aa9fe-913d-4ede-af58-73e332dcac8d     FALSE  0    0    0
1b1d4ff4-f27b-4c99-8bd7-da8f1a74051a      TRUE  3    0    0
67e605ee-a4d8-4c46-ae50-893f31e13963     FALSE  0    0    0
f152dc14-a3a0-4258-bece-69a3ee4c2de8     FALSE  0    0    0
bc714241-8edc-4ce3-8714-aa0b51f98fdf     FALSE  0    0    0
82d60869-5ada-4d49-b76a-309b09666584     FALSE  0    0    0
2b74a015-3873-4c56-9928-ea80c58b2787     FALSE  0    0    0
f33959b4-dbec-11d2-895b-00c04f79ab69     FALSE  0    0    0
1540ff4c-3fd7-4bba-9938-1d1bf31573a7     FALSE  0    0    0
1c83b2fc-c04f-11d1-8afc-00c04fc21914     FALSE  0    0    0
3121cf5d-c5e6-4f37-be86-57083590c333     FALSE  0    0    0
24db8964-e6bc-11d1-916a-0000f8045b04     FALSE  0    0    0
6a187a25-2325-45f4-a928-b554329ebd51     FALSE  0    0    0
8e598056-8993-11d2-819e-0000f875a064     FALSE  0    0    0
f2969c49-b484-4485-b3b0-b908da73cebb     FALSE  0    0    0
9474a749-a98d-4f52-9f45-5b20247e4f01     FALSE  0    0    0
cc85922f-db41-11d2-9244-006008269001     FALSE  0    0    0
c92cf544-91b3-4dc0-8e11-c580339a0bf8     FALSE  0    0    0
bba3add2-c229-4cdb-ae2b-57eb6966b0c4     FALSE  0    0    0
bda92ae8-9f11-4d49-ba1d-a4c2abca692e     FALSE  0    0    0
8fc7e81a-f733-42e0-9708-cfdae07ed969     FALSE  0    0    0
cddc01e2-fdce-479a-b8ee-3c87053fb55e     FALSE  0    0    0
b40aef77-892a-46f9-9109-438e399bb894     FALSE  0    0    0
fc4b0d39-e8be-4a83-a32f-c0c7c4f61ee4     FALSE  0    0    0
fc570986-5967-4641-a6f9-05291bce66c5     FALSE  0    0    0
39a7b5e0-be85-47fc-b9f5-593a659abac1     FALSE  0    0    0
dab01d4d-2d48-477d-b1c3-daad0ce6f06b     FALSE  0    0    0
bca7bd7f-b0bf-4051-99f4-03cfe79664c1     FALSE  0    0    0
d58c126f-b309-11d1-969e-0000f875a5bc     FALSE  0    0    0
d58c126e-b309-11d1-969e-0000f875a5bc     FALSE  0    0    0
58db8e03-0537-45cb-b29b-597f6cbebbfd     FALSE  0    0    0
cd079d47-329d-4dc5-881c-cb28bb80a9a0     FALSE  0    0    0
47666aa4-63f7-497d-b245-68fecab82f17     FALSE  0    0    0
47666aa4-63f7-497d-b122-68fecab82f17     FALSE  0    0    0
27246e9d-b4df-4f20-b969-736fa49ff6ff     FALSE  0    0    0
f96abc17-6a5e-4a49-a3f4-a2a86fa03846     FALSE  0    0    0
544d4c9d-942c-46d5-bf50-df5cd9524a50     FALSE  0    0    0
cb5b2c18-ad73-4ebf-8af1-73b30b885030     FALSE  0    0    0

C:\ETW Tools\x86>_
```

Here is a sample using Logman to get the available providers:



```
C:\Users\rguthrie>logman query providers

Provider                                 GUID
-------------------------------------------------------------------------------
.NET Common Language Runtime             {E13C0D23-CCBC-4E12-931B-D9CC2EEE27E4}
ACPI Driver Trace Provider               {DAB01D4D-2D48-477D-B1C3-DAAD0CE6F06B}
Active Directory Domain Services: SAM    {8E598056-8993-11D2-819E-0000F875A064}
Active Directory: Kerberos Client        {BBA3ADD2-C229-4CDB-AE2B-57EB6966B0C4}
Active Directory: NetLogon               {F33959B4-DBEC-11D2-895B-00C04F79AB69}
ASP.NET Events                           {AFF081FE-0247-4275-9C4E-021F3DC1DA35}
ATA Port Driver Tracing Provider         {D08BD885-501E-489A-BAC6-B7D24BFE6BBF}
Audio_AudioTrace                         {E27950EB-1768-451F-96AC-CC4E14F6D3D0}
AuthFw NetShell Plugin                   {935F4AE6-845D-41C6-97FA-380DAD429B72}
BFE Trace Provider                       {106B464A-8043-46B1-8CB8-E92A0CD7A560}
BITS Service Trace                       {4A8AAA94-CFC4-46A7-8E4E-17BC45608F0A}
Certificate Services Client CredentialRoaming Trace {EF4109DC-68FC-45AF-B329-CA2
825437209}
Certificate Services Client Trace        {F01B7774-7ED7-401E-8088-B576793D7841}
Classpnp Driver Tracing Provider         {FA8DE7C4-ACDE-4443-9994-C4E2359A9EDB}
Common Log (CLFS)                        {9690A7F3-49F1-4529-B3D6-57797343AC75}
Critical Section Trace Provider          {3AC66736-CC59-4CFF-8115-8DF50E39816B}
Disk Class Driver Tracing Provider       {945186BF-3DD6-4F3F-9C8E-9EDD3FC9D558}
Downlevel IPsec API                      {94335EB3-79EA-44D5-8EA9-306F49B3A041}
Downlevel IPsec NetShell Plugin          {E4FF10D8-8A88-4FC6-82C8-8C23E9462FE5}
Downlevel IPsec Policy Store             {94335EB3-79EA-44D5-8EA9-306F49B3A070}
```

Not all of these providers are useful to collect logging information. You will need to work with Microsoft Product Support to determine the correct logging parameters.

You will need to start the trace collection, execute a scenario that reproduces the issue and then stop the trace. The resulting .etl file should then be sent to Microsoft for analysis.

Here is an example of a typical command with tracelog.exe and logman.exe

tracelog -start LoggingInstanceName -guid SMB-Guids.txt -f OutPutLogFile.log -level 4 -flags 2

Perform the task to generate the desired behavior

tracelog – stop LoggingInstanceName

This example starts a trace log and passes in a file, SMB-Guids.txt, with a list of GUIDs for the event providers we want to log. The logging level is set to 4 for information and flags is set to 2 which is used to refine the amount of logged events.

The text file SMB-Guids.txt would look as follows (note that each GUID is listed on a separate line):

8fc7e81a-f733-42e0-9708-cfdae07ed969 MRxSmb
cddc01e2-fdce-479a-b8ee-3c87053fb55e Rdbss

logman.exe query providers

This will query for a full list of ETW providers available on the system.

Applicable scenarios for interoperability

- Diagnosing issues related to ADSI (many Active Directory tools use this as a data provider for querying the directory store)

- Security related issues (NTLM, NTLMv2, Kerberos, SPNegotiate)

Reference Information

Trace levels

Trace Level	Trace Constant	Description
NONE	0	Tracing is not on
FATAL	1	Abnormal exit or termination
ERROR	2	Severe errors that need logging
WARNING	3	Warnings such as allocation failure
INFORMATION	4	Includes non-error cases such as Entry-Exit

ETW Examples

Some examples of ETW tracing usage can be found here:

- Event Tracing in ADSI
- Security Account Manager (SAM)

Useful Reference Links

The following are some useful reference links:

- Disable signed or encrypted LDAP traffic
- Client, service, and program incompatibilities that may occur when you modify security settings and user rights assignments
- LDAP signing changes for Active Directory administrative tools in Windows 2000 Server Service Pack 4

ETW Related Blogs

Some additional blog links for more information on ETW can be found here:

- Under The Hood - Matt Pietrek
- New Tools for Event Management in Windows Vista
- FTP and ETW Tracing
- Mike Devlin – Tracelog info