TZWorks® Trace Event Log Analysis (*tela*) Users Guide

Abstract

tela is a command-line tool that parses Windows Trace Log files into CSV type records. Other capabilities: include generating statistics on a trace logs, creating a subset trace log, and enumeration of trace log providers. ***tela*** runs on Windows, Linux and Mac OS-X.

Document applies to v0.31 of ***tela***

Updated: Apr 25, 2025

Table of Contents

TZWorks® Trace Event Log Analysis (*tela*) Users Guide

Copyright © *TZWorks LLC*
Contact Information: info@tzworks.com

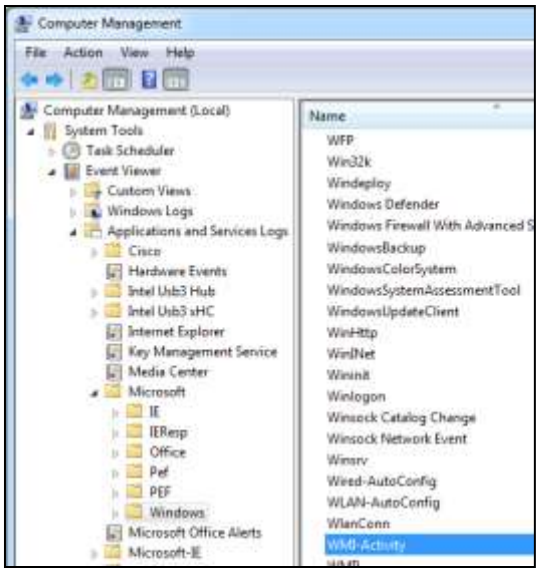
1 Introduction

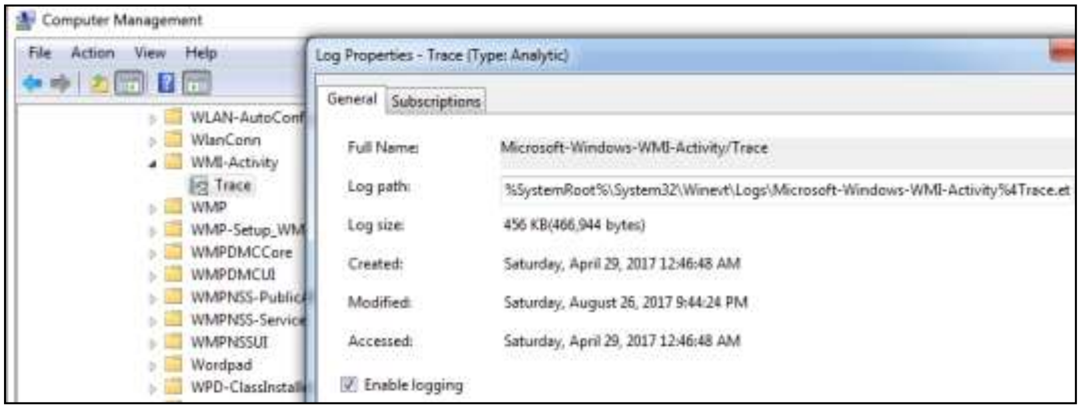
Event Tracing for Windows or ETW, is a built-in, logging and diagnostic framework available to all. It can be dynamically enabled and the data it generates can be logged to a file or consumed in real time by another application. It can be used for performance analysis, general debugging, or in our case, for forensics purposes.

ETW was first introduced in Window 2000. It started as a modest set of providers and was expanded with each new version of Windows. With Windows 10, the number of providers grew to over 1000. Starting with Win10, TraceLogging was added to ETW to allow for event tracing for user-mode applications and kernel-mode drivers.

2 Background Information

To enable logging in Windows, one can do it a number of ways. An easy way is via the Windows Computer Management console in the “Applications and Services Logs” of the Event Viewer subsection. One needs to ensure the “Show Analytic and Debug Logs” is enabled. After that many different apps and services can have Trace logs enabled. Below is an example of doing it on WMI-Activity.





Full Name: Microsoft-Windows-WMI-Activity/Trace
Log path: %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-WMI-Activity%4Trace.et
Log size: 456 KB(466,944 bytes)
Created: Saturday, April 29, 2017 12:46:48 AM
Modified: Saturday, August 26, 2017 9:44:24 PM
Accessed: Saturday, April 29, 2017 12:46:48 AM
☑ Enable logging

Once enabled, the system will start populating the Event Trace Log (ETL) file with trace data. The properties dialog will show where the trace data will be logged and allow one to set other parameters as well, such as max log size, whether to overwrite older trace data once the max sized is reached, subscribe to received trace events from another computer, etc.

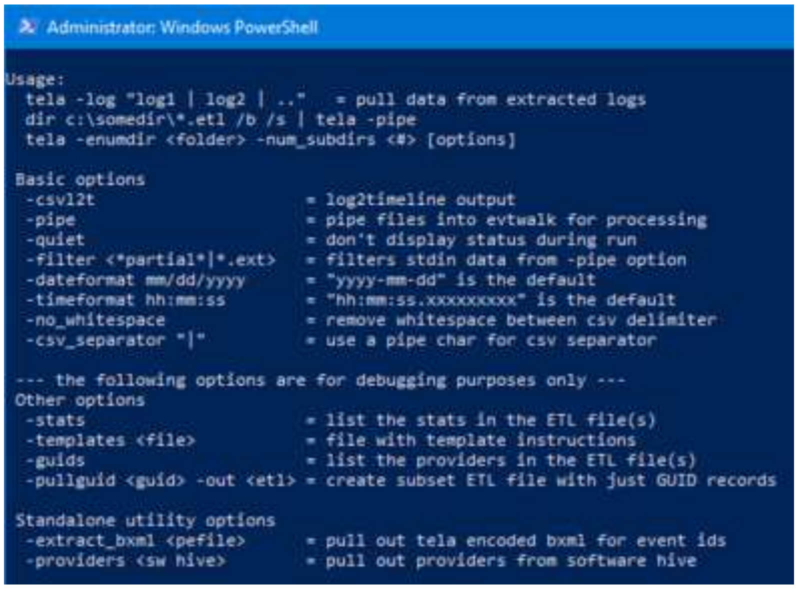
2.1 Using the Event Log Viewer in Windows to view the ETL data

The Microsoft's Event Viewer will allow one to view ETL files, with some caveats. On our test system, it would recognize most, but there were some that it would not. Of the data that was recognized, it was primarily the header data and not the payload data. To see the details of the payload data in the ETL there was another tool provided called the Microsoft Message Analyzer. From the online help for the tool, it "is a new tool for capturing, displaying and analyzing protocol messaging traffic, events and other system or application messages in network troubleshooting and other diagnostic scenarios… [It] also enables you to load, aggregate, and analyze data from log and saved trace files."

These were some of the tools used to design ***tela,*** and were a good way to validate the data ***tela*** produced. So our goal was at least to have ***tela*** parse the same ETL data that the Microsoft Message Analyzer did. In all the cases we tested ***tela*** was able to parse the header for all the records it encountered. Keep in mind, our ETL test data is limited. Where ***tela*** falls short is parsing the payload data in the records. More discussion about this is in the section on "*Mapping Provider/Event ID to PE Resource Data*".

3 How to Use the *tela* Tool

To just parse a single ETL file, one uses the ***-log*** option with the argument of the path of the ETL file. This will generate a single record per line where each field is delimited by a '|' (pipe) character. Redirecting the output to a file is recommended, since the widths of the records are large. The other basic option is to enumerate a set of files in a folder and pipe them into tela via standard input.   These options are the first two shown in the menu below.



```
Usage:
  tela -log "log1 | log2 | .."    = pull data from extracted logs
  dir c:\somedir\*.etl /b /s | tela -pipe
  tela -enumdir <folder> -num_subdirs <#> [options]

 Basic options
  -csvl2t                       = log2timeline output
  -pipe                         = pipe files into evtwalk for processing
  -quiet                        = don't display status during run
  -filter <*partial*|*.ext>     = filters stdin data from -pipe option
  -dateformat mm/dd/yyyy        = "yyyy-mm-dd" is the default
  -timeformat hh:mm:ss          = "hh:mm:ss.xxxxxxxxx" is the default
  -no_whitespace                = remove whitespace between csv delimiter
  -csv_separator "|"            = use a pipe char for csv separator

 --- the following options are for debugging purposes only ---
 Other options
  -stats                        = list the stats in the ETL file(s)
  -templates <file>             = file with template instructions
  -guids                        = list the providers in the ETL file(s)
  -pullguid <guid> -out <etl> = create subset ETL file with just GUID records

 Standalone utility options
  -extract_bxml <pefile>        = pull out tela encoded bxml for event ids
  -providers <sw hive>          = pull out providers from software hive
```

The next section of includes a number of basic options used to change the formatting of the output or to filter out files when piping a set of files from standard input. These options are very similar, if not the same, as other tools in *TZWorks*.

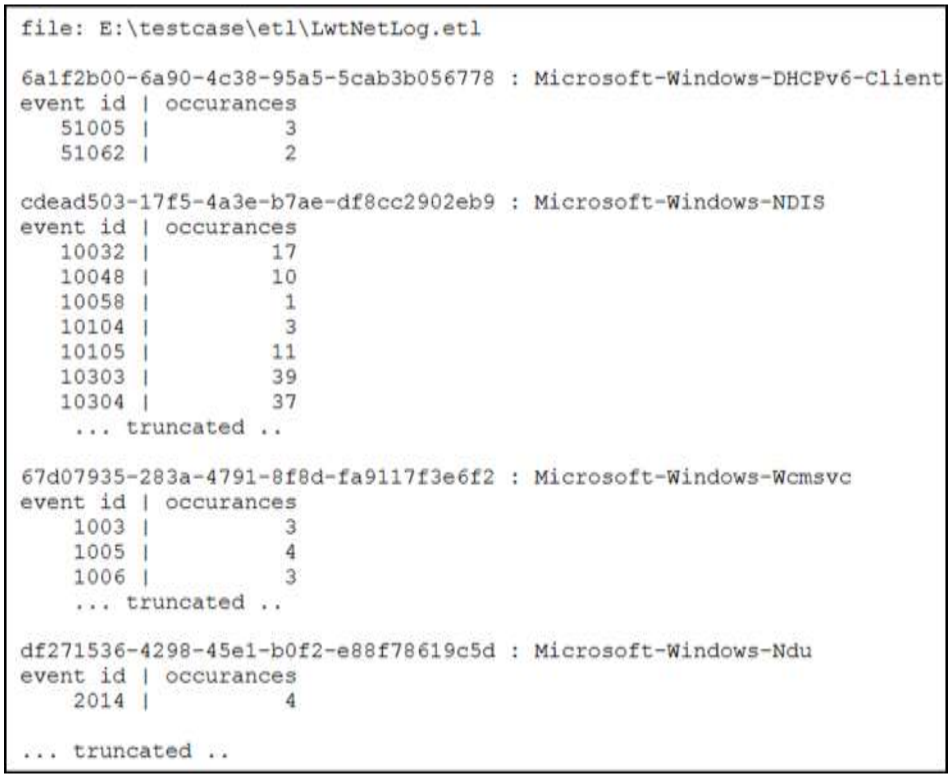
The last two sections are options used for reversing and/or debugging purposes only, and has been made available for the analyst to use at their discretion.   These options include: (a) gathering statistics on the ETL file, (b) using an external template file (which is for extensibility and testing purposes), (c) listing out provider GUIDs in an ETL file, (d) creating a separate ETL file from records extracted from a specified provider, and (e) using other TZWorks tools to extract pertain data from artifacts to assist in the analysis of an ETL file.

3.1 Quick-look Report for a ETL file

During the analysis and design ***tela*** it was necessary to determine which providers and event ID's were present in a specific ETL file. This was helpful in going after metadata that was being reversed and that would only be present for certain combinations of provider/event ID's. For this use-case, the ***-stats*** option was added to the design. Using this option will cause ***tela*** to histogram the events by their respective provider.

Below is a sample output using this option. Various sections are truncated to not take up too much space in the document, but enough to get the point across as to what the output looks like. If the provider name for the provider GUID is known by ***tela***, it be outputted as shown below. For each provider, a set of event ID's are displayed along with the number of occurrences.

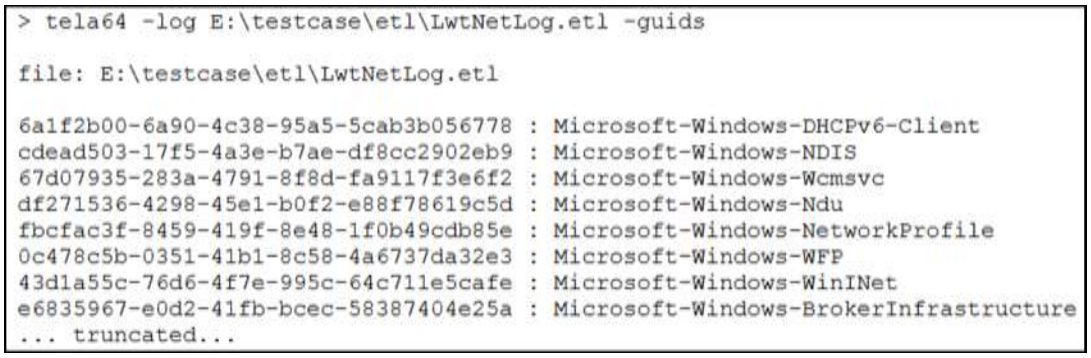
```
tela64 -log E:\testcase\etl\LwtNetLog.etl -stats > out.txt
```



```
file: E:\testcase\etl\LwtNetLog.etl

6a1f2b00-6a90-4c38-95a5-5cab3b056778 : Microsoft-Windows-DHCPv6-Client
event id | occurances
   51005 |          3
   51062 |          2

cdead503-17f5-4a3e-b7ae-df8cc2902eb9 : Microsoft-Windows-NDIS
event id | occurances
   10032 |         17
   10048 |         10
   10058 |          1
   10104 |          3
   10105 |         11
   10303 |         39
   10304 |         37
    ... truncated ..

67d07935-283a-4791-8f8d-fa9117f3e6f2 : Microsoft-Windows-Wcmsvc
event id | occurances
    1003 |          3
    1005 |          4
    1006 |          3
    ... truncated ..

df271536-4298-45e1-b0f2-e88f78619c5d : Microsoft-Windows-Ndu
event id | occurances
    2014 |          4

... truncated ..
```

Without this option, one would need to wade through many lines (typically 1000's) of output to identify whether a specific pair of provider to event ID was present. So, even though it was added as an aid in the development of ***tela***, the option was left in since it can also be useful to the analyst, as a quick look report.

As a subset of this option there is also the ***-guids*** option, which will just list the provider GUID and associated name, if available. The syntax is the same as the previous command, but substituting word ***-stats*** with ***-guids***.



```
> tela64 -log E:\testcase\etl\LwtNetLog.etl -guids

file: E:\testcase\etl\LwtNetLog.etl

6a1f2b00-6a90-4c38-95a5-5cab3b056778 : Microsoft-Windows-DHCPv6-Client
cdead503-17f5-4a3e-b7ae-df8cc2902eb9 : Microsoft-Windows-NDIS
67d07935-283a-4791-8f8d-fa9117f3e6f2 : Microsoft-Windows-Wcmsvc
df271536-4298-45e1-b0f2-e88f78619c5d : Microsoft-Windows-Ndu
fbcfac3f-8459-419f-8e48-1f0b49cdb85e : Microsoft-Windows-NetworkProfile
0c478c5b-0351-41b1-8c58-4a6737da32e3 : Microsoft-Windows-WFP
43d1a55c-76d6-4f7e-995c-64c711e5cafe : Microsoft-Windows-WinINet
e6835967-e0d2-41fb-bcec-58387404e25a : Microsoft-Windows-BrokerInfrastructure
... truncated...
```

3.2 Parsing out ETL records

When pulling out the records of an ETL file, one can either use the ***-log*** option and explicitly pass in the path of the file, or use the ***-pipe*** option and use standard input. Either way, the record output is the same, but the second approach will have a small header per file added, so the output is clear which file is associated with which records.

There are a few things to note about the records contained in an ETL file.

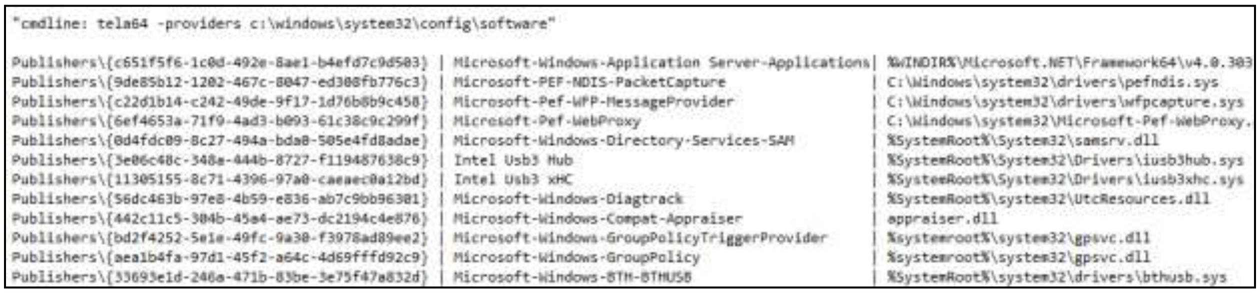
- As shown in the previous example, more than one provider can be embedded in an ETL file.
- Given that each provider can have their own set of version changes and templates, this means that each record from one provider to the next may or may not have the same structure for the header. Different structures for the headers means the parser dynamically switches between each header type encountered. This makes rendering the output of the data more complex since the field from one header may not be in another header. ***tela*** does a good job at aligning the common fields appropriately. For the differing data, ***tela*** uses a single field that is expressed in a quasi JSON output to render these unique fields.
- Each record in the ETL may or may not have payload data in addition to its header data. The payload data varies based on the event ID associated with that record. For any available event template data that was available during the design ***tela***, that payload data structure was included into ***tela***, and consequently will have the appropriate field names associated with the values in the payload as part of the output. If the template data was not available to us, then ***tela*** will try to figure out which data are strings, values, etc. This latter case, of course, would be a guess, and ***tela***, identifies this data with labels prepended with the letters *guess_*.
- If the ETL file uses any type of compression, then ***tela*** will not be able to parse it, since no decompression logic was added to the tool. Specifically, if you use a tool to copy ETL file off a disk in a raw manner (eg. via cluster reads) and the data is using the NTFS compression, ***tela*** will not be able to parse that data.

- The timestamps for ETL records can use more than one format, ranging from FILETIME data (which is normal), QPC, or CPU cycle counter. The first is normal and easy to convert to UTC. The latter two, require some additional translation and are dependent on other parameters to convert them to UTC. This is because they are not synchronized to any external time reference. Therefore to retrieve time stamps that can be synchronized to an external time reference, one needs to use other parameters such as the frequency of the Performance Counter and CPU speed. Depending on whether this other data is present dictates whether the output displays the time in UTC or tick count.

With the limitations discussed above, tela is pretty robust at parsing ETL files, in general.

3.3 Searching for Providers

All the ETL data is based on a specific provider and how that provider wishes to format the data. The provider data in each record are in terms of GUIDs. To understand what a GUID value translates to, one can look to the Software registry hive (or use Windows *Powershell* to enumerate the providers). Alternatively, ***tela*** can pull the data from the registry for you via the ***-providers*** option. ***tela*** will go off and spawn ***cafae*** to pull the data from the subkey: [SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers]. Below is an example of doing this on one of our test systems:



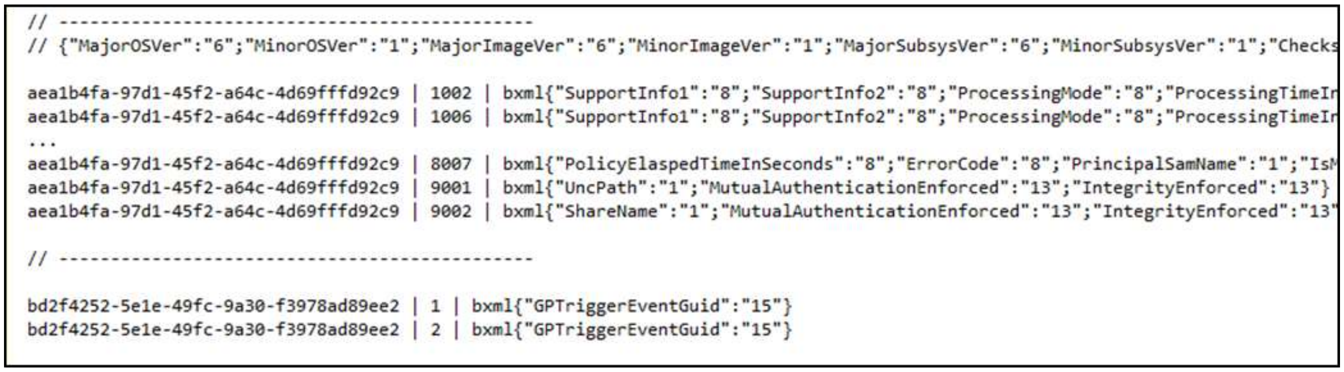
```
"cmdline: tela64 -providers c:\windows\system32\config\software"

Publishers\{c651f5f6-1c0d-492e-8ae1-b4efd7c9d503} | Microsoft-Windows-Application Server-Applications| %WINDIR%\Microsoft.NET\Framework64\v4.0.303
Publishers\{9de85b12-1202-467c-8047-ed308fb776c3} | Microsoft-PEF-NDIS-PacketCapture                 | C:\Windows\system32\drivers\pefndis.sys
Publishers\{c22d1b14-c242-49de-9f17-1d76b8b9c458} | Microsoft-Pef-WFP-MessageProvider                | C:\Windows\system32\drivers\wfpcapture.sys
Publishers\{6ef4653a-71f9-4ad3-b093-61c38c9c299f} | Microsoft-Pef-WebProxy                           | C:\Windows\system32\Microsoft-Pef-WebProxy.
Publishers\{0d4fdc09-8c27-494a-bda0-505e4fd8adae} | Microsoft-Windows-Directory-Services-SAM         | %SystemRoot%\System32\samsrv.dll
Publishers\{3e06c48c-348e-444b-8727-f119487638c9} | Intel Usb3 Hub                                   | %SystemRoot%\System32\Drivers\iusb3hub.sys
Publishers\{11305155-8c71-4396-97a0-caeaec0a12bd} | Intel Usb3 xHC                                   | %SysteeRoot%\System32\Drivers\iusb3xhc.sys
Publishers\{56dc463b-97e8-4b59-e836-ab7c9bb96301} | Microsoft-Windows-Diagtrack                      | %SystemRoot%\system32\UtcResources.dll
Publishers\{442c11c5-304b-45a4-ae73-dc2194c4e876} | Microsoft-Windows-Compat-Appraiser               | appraiser.dll
Publishers\{bd2f4252-5e1e-49fc-9a30-f3978ad89ee2} | Microsoft-Windows-GroupPolicyTriggerProvider     | %systemroot%\system32\gpsvc.dll
Publishers\{aea1b4fa-97d1-45f2-a64c-4d69fffd92c9} | Microsoft-Windows-GroupPolicy                    | %systemroot%\system32\gpsvc.dll
Publishers\{33693e1d-246a-471b-83be-3e75f47a832d} | Microsoft-Windows-BTH-BTHUSB                     | %SystemRoot%\system32\drivers\bthusb.sys
```

The first column above shows the provider GUID mapped to the provider name in the second column. The 3rd column is also important, and it is the portable executable (PE) file that contains the resource that holds the template on how to translate the ETL payload data.

3.4 Mapping Provider/Event ID to PE Resource Data

Each record in the ETL file will have a set of common structure for the header data and set of bytes for the payload data. To make sense of the payload data, one needs to extract the template that is stored in the PE resource data associated with the provider and event identifier. This section provides a quick set of procedures to do just that using ***tela*** and other *TZWorks* tools.

In the previous section, it was discussed how to pull the PE resource associated to a provider GUID or name. Referring to the output in the previous section, the third column is the critical data that is needed.   If we take one of the resources on the right, such as the one for *GroupPolicy* [*gpsvc.sys]*, one can look at the template data that is present in this resource. This can be done via the ***-extract_bxml*** option and passing in the PE resource as an argument. When this is invoked, ***tela*** will go off and spawn ***pescan*** to pull the *WEVT_TEMPLATE* data from the specified PE file, and render it like so:



```
// ---------------------------------------------
// {"MajorOSVer":"6";"MinorOSVer":"1";"MajorImageVer":"6";"MinorImageVer":"1";"MajorSubsysVer":"6";"MinorSubsysVer":"1";"Checks

aea1b4fa-97d1-45f2-a64c-4d69fffd92c9 | 1002 | bxml{"SupportInfo1":"8";"SupportInfo2":"8";"ProcessingMode":"8";"ProcessingTimeIn
aea1b4fa-97d1-45f2-a64c-4d69fffd92c9 | 1006 | bxml{"SupportInfo1":"8";"SupportInfo2":"8";"ProcessingMode":"8";"ProcessingTimeIn
...
aea1b4fa-97d1-45f2-a64c-4d69fffd92c9 | 8007 | bxml{"PolicyElaspedTimeInSeconds":"8";"ErrorCode":"8";"PrincipalSamName":"1";"IsM
aea1b4fa-97d1-45f2-a64c-4d69fffd92c9 | 9001 | bxml{"UncPath":"1";"MutualAuthenticationEnforced":"13";"IntegrityEnforced":"13"}
aea1b4fa-97d1-45f2-a64c-4d69fffd92c9 | 9002 | bxml{"ShareName":"1";"MutualAuthenticationEnforced":"13";"IntegrityEnforced":"13"

// ---------------------------------------------

bd2f4252-5e1e-49fc-9a30-f3978ad89ee2 | 1 | bxml{"GPTriggerEventGuid":"15"}
bd2f4252-5e1e-49fc-9a30-f3978ad89ee2 | 2 | bxml{"GPTriggerEventGuid":"15"}
```

Starting with the second line that has a double forward slash is the PE version and which OS it targets. This is important, since different versions may have different provider *WEVT_TEMPLATE* data. This data is then used to map into which OS the ETL file came from.   Then there is a list of records, where a record starts with the provider GUID, then the event ID, and finally the template data when the numbers in this data use the BXML schema. Towards the end of the output there are two other records where there is listed another provider GUID and its template data. This means each PE *WEVT_TEMPLATE* resource can contain more than 1 provider type data.

While the above output is somewhat compact, it is useful to us internally to build templates to allow ***tela*** to parse various provider data. Showing it here is primarily for illustrative purposes help the reader understand the process required to parse the payload data in the ETL records.

3.5 Using Templates to assist *tela*

An internal option is exposed that allows for users to point ***tela*** to an external template type file, via the syntax ***-template <file>***. The purpose of this option was to allow the tool have parsing extensibility as new providers become of interest.

Although this option is functional, the generation of these templates was meant to be done internally until this functionality matures out of the beta phase. This is because the generation of these template files requires one to have good knowledge of the *WEVT_TEMPLATE* data from the provider. Further, this requires one to understand the output of the raw *WEVT_TEMPLATE* data, and how to apply that data to parse ETL structures.

4 Available Options

Option	Description
-log	Identify which event log(s) to operate on. The syntax is: ***-log <ETL to analyze>***. To operate one more than one at a time, use: ***-log "<ETL1> | <ETL2> | ..."***
-csvl2t	Outputs the data fields in accordance with the log2timeline format.
-pipe	This option allows one to parse in multiple tracelogs from standard input in one session.
-enumdir	Experimental. Used to process files within a folder and/or subfolders. Each file is parsed in sequence. The syntax is ***-enumdir <folder> -num_subdirs <#>***.
-filter	Filters data passed in via stdin via the ***-pipe*** or ***-enumdir*** options. The syntax is ***-filter <"*.ext | *partialname* | ...">***. The wildcard character '*' is restricted to either before the name or after the name.
-quiet	This option is tells the app not to display progress status during a run.
-no_whitespace	This option will remove all white space between the field value and the delimiter separator.
-csv_separator	Used in conjunction with the ***-csv*** option to change the CSV separator from the default comma to something else. Syntax is ***-csv_separator "|"*** to change the CSV separator to the pipe character.
-dateformat	Output the date using the specified format. Default behavior is *-dateformat "yyyy-mm-dd"*. Using this option allows one to adjust the format to mm/dd/yy, dd/mm/yy, etc. The restriction with this option is the forward slash (/) or dash (-) symbol needs to separate month, day and year and the month is in digit (1-12) form versus abbreviated name form.
-timeformat	Output the time using the specified format. Default behavior is ***-timeformat "hh:mm:ss.xxxxxxxxx"*** One can adjust the format to microseconds, via ***"hh:mm:ss.xxxxxx"*** or millisecs, via ***"hh:mm:ss.xxx"***, or no fractional seconds, via ***"hh:mm:ss"***. The restrictions with this option is that a colon (:) symbol needs to separate hours, minutes and seconds, a period (.) symbol needs to separate the seconds and fractional seconds, and the repeating symbol 'x' is used to represent number of fractional seconds.
-templates	Use the specified file that stores BXML templates to parse ETL file. The syntax is: ***-templates <file>***.
-pullguid	Create a subset ETL file with just the specified GUID (provider) records. The syntax is: ***-pullguid <guid> -out <subset ETL file>***.

-guids	List all the providers (guids) used in the ETL file(s)
-stats	List all the providers (guids) and the Event IDs used in the ETL file(s). The Event IDs will be listing in terms of occurrence (like a histogram).
-extract_bxml	Extract the BXML Templates from the portable executable (PE) file. The syntax is: ***-extract_bxml <pefile>***. This option relies on the presence of ***pescan*** to be in the same directory as ***tela***.
-providers	Extract the providers and their associate guid and resource file from the specified Software hive. The syntax is: ***-providers <sw hive>***. This option relies on the presence of ***cafae*** to be in the same directory as ***tela***.
-utf8_bom	All output is in Unicode UTF-8 format. If desired, one can prefix an UTF-8 *byte order mark* to the CSV output using this option.

5 Common locations for ETL files

System volume location	ETL file
**%ProgramData%\Microsoft\Windows\WER\ReportQueue\*\\**	{guid}-WER-MMDDYYYY-####
%ProgramData%\Microsoft\Windows\wfp\	wfpdiag
%ProgramData%\USOShared\Logs\	NotificationUxBroker, UpdateSessionOrchestration
%LocalAppData%\Packages\Microsoft.CommsPhone_..\LocalCache\	CallsAppLog, CallsBackgroundTaskLog,
%LocalAppData%\Packages\Microsoft.Messaging_...\LocalCache\	MessagingBackgroundTaskLog
%LocalAppData%\Packages\Microsoft.Windows.Photos_...\LocalState\	PhotosAppTracing
**%LocalAppData%\Diagnostics\**\\**	{guid}.Diagnose, {guid}.Repair, {guid}.Verify
**%LocalAppData%\ElevatedDiagnostics\**\\**	{guid}.Diagnose.Admin, {guid}.Repair.Admin, {guid}.Verify.Admin
%LocalAppData%\Microsoft\Office\	OTeleData
%LocalAppData%\Microsoft\OneDrive\logs\Personal\	TraceCurrent, TraceArchive
%LocalAppData%\Microsoft\Windows\Explorer\	ExplorerStartupLog
%LocalAppData%\Temp	{guid}.Repair, {guid}.Diagnose, {guid}.Verify
%UserProfile%\Tracing\WPPMedia\	Skype_MediaStack
%systemroot%\Logs\dosvc\	dosvc.YYYYMMDD_xxxxx
%systemroot%\Logs\NetSetup\	service.x
%systemroot%\Logs\SIH\	SIH.YYYYMMDD.xxxxx
%systemroot%\Logs\SystemRestore\	Restore.UI
%systemroot%\Logs\WindowsBackup\	WBEngine.x
%systemroot%\Logs\WindowsUpdate\	WindowsUpdate.YYYYMMDD.xxxxx
%systemroot%\Panther\	setup
%systemroot%\Performance\WinSAT\DataStore\	winsat
%systemroot%\security\logs\	SceSetupLog
**%systemroot%\SoftwareDistribution\Download\**\\**	WindowsUpdate.YYYYMMDD.xxxx, SceSetupLog, LwtNetLog, Wifi, EtwRTDiagLog, EtwRTEventLog-Application, EtwRTEventLog-System, ShutdownCKCL, BootCKCL, SecondaryLogonCKCL, wfpdiag, KnobsCsp
%systemroot%\System32\LogFiles\WMI\	Wifi, FamilySafetyAOT, LwtNetLog, ..
%systemroot%\System32\LogFiles\WMI\RtBackup\	EtwRTDiagLog, EtwRTEventLog-System, EtwRTEventLog-Application, EtwRTUBPM,

	EtwRTEventlog-Security, EtwRTDefenderAuditLogger, EtwRTWFP-IPsec Diagnostics, EtwRTDiagtrack-Listener, EtwRTDefenderApiLogger, ...
%systemroot%\System32\NDF\	eventlog
%systemroot%\System32\SleepStudy\	sleepstudy-trace-yyyy-mm-dd-hh-mm-ss, SleepStudyTraceSession
%systemroot%\System32\WDI\{guid}\{guid}	snapshot
%systemroot%\System32\WDI\LogFiles\	SecondaryLogonCKCL, ShutdownCKCL, BootCKCL, ...
%systemroot%\System32\winevt\Logs\	AirSpaceChannel, ...

6 Authentication and the License File

This tool has authentication built into the binary. The primary authentication mechanism is the digital X509 code signing certificate embedded into the binary (Windows and macOS).

The other mechanism is the runtime authentication, which applies to all the versions of the tools (Windows, Linux and macOS). The runtime authentication ensures that the tool has a valid license. The license needs to be in the same directory of the tool for it to authenticate. Furthermore, any modification to the license, either to its name or contents, will invalidate the license.

7 References

1. EventTracing on MSDN website and related articles - https://msdn.microsoft.com/en-us/library/windows/desktop/bb968803(v=vs.85).aspx
2. Microsoft Message Analyzer, version 1.4 (build 4.0.8112.0)
3. Microsoft Event Viewer (Windows 7, 8, and 10)
4. Microsoft TraceView Tool (Windows SDK)