Microsoft

Microsoft Build 2017

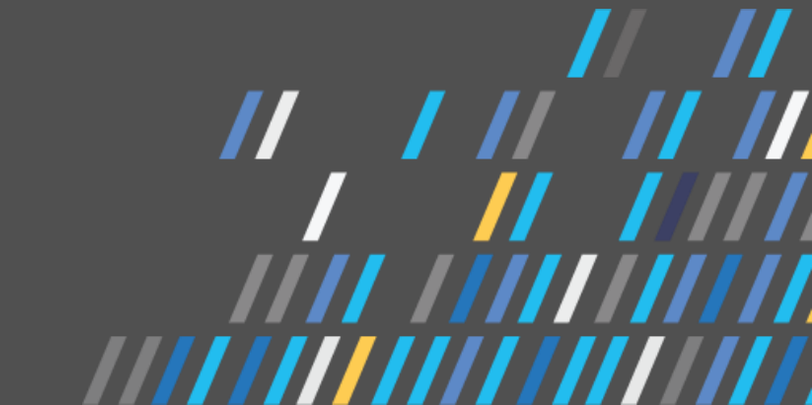
Microsoft

P4099

Production Tracing with Event Tracing for Windows (ETW)

Doug Cook
Software Engineer

#MSBuild



Tracing – getting info from your program

- printf, Console.WriteLine
  - Great for console tools or during development.
  - Not great for GUI apps, web apps, services, drivers.
  - Not great in production/retail environments.
- OutputDebugString, DbgPrint, Debug.WriteLine
  - Great during development.
  - Don’t use in production (performance impact, no filtering).
- Log files
  - Great for low-volume information that must be kept long-term.
  - Hard to manage for detailed (high-volume) or diagnostic data.

What is ETW?

- Event Tracing for Windows.
- Routes information from your program to an analysis tool.
  - Sends data to log file, memory buffer, or real-time consumer.
- Works for drivers, services, and apps.
- Separation of concerns between event producer/consumer.
- Development, test, and production scenarios.
  - Tracing is disabled by default
  - Almost no performance impact when tracing disabled.
  - Low impact (non-blocking) when tracing enabled.
  - Powerful filtering (change filters without restarting app).

Demo: ETW Capture

ETW scenarios

- Debugging.
  - Example: trace AddRef and Release in a COM object.
- Field diagnostics.
  - Events can be temporarily enabled in production to diagnose issues.
- Flight recorder.
  - Always collect app events into a process-private circular buffer.
  - If an error occurs (e.g. unexpected exception), flush buffer to disk and save it for investigation (include in bug report).
- Log file.
  - Always collect important app events into a process-private log file.

Demo: TraceLogging C++

Demo: TraceLogging C++

- Use Windows 10 SDK.
- #include <TraceLoggingProvider.h>
  - Read the comments in the header for more information.
  - Requires Windows 8 by default; optionally compatible back to Vista. #define _WIN32_WINNT _WIN32_WINNT_VISTA
- TRACELOGGING_DEFINE_PROVIDER
  - Recommended: use EtwGuid tool to generate provider GUID. https://blogs.msdn.microsoft.com/dcook/2015/09/08/etw-provider-names-and-guids/
- TraceLoggingRegister/TraceLoggingUnregister
- TraceLoggingWrite

Demo: TraceLogging .NET

Demo: TraceLogging .NET

- Use .NET 4.6 or later.
  - For compatibility with earlier frameworks, use NuGet package "EventSource Redist".
- Define a global EventSource object.
  - public static readonly EventSource MyLogger =
    new EventSource("ProviderName");
- Call the Write method.
  - MyLogger.Write(
    "EventName",
    new { FieldName1 = Value1, FieldName2 = Value2 });

Getting started with ETW

Provider libraries:

- C/C++: TraceLoggingProvider.h
- .NET: EventSource
- Windows Framework: LoggingChannel

Consumer tools:

- GUI trace control: traceview, perfview
  - Traceview updated in Windows 10 Creators Edition SDK.
- GUI trace analysis: traceview, perfview, WPA
- Command-line trace control: tracelog, xperf, WPR
- Command-line trace analysis: tracefmt, tracerpt

ETW Frameworks

ETW framework

- Language for describing provider and event characteristics (metadata).
  - Provider name, event name, severity level, field names, field types, etc.
- Code generation.
  - First layer of event filtering (provider, event level, event keywords).
  - Packs event data.
- Process for getting metadata from you to the decoder.

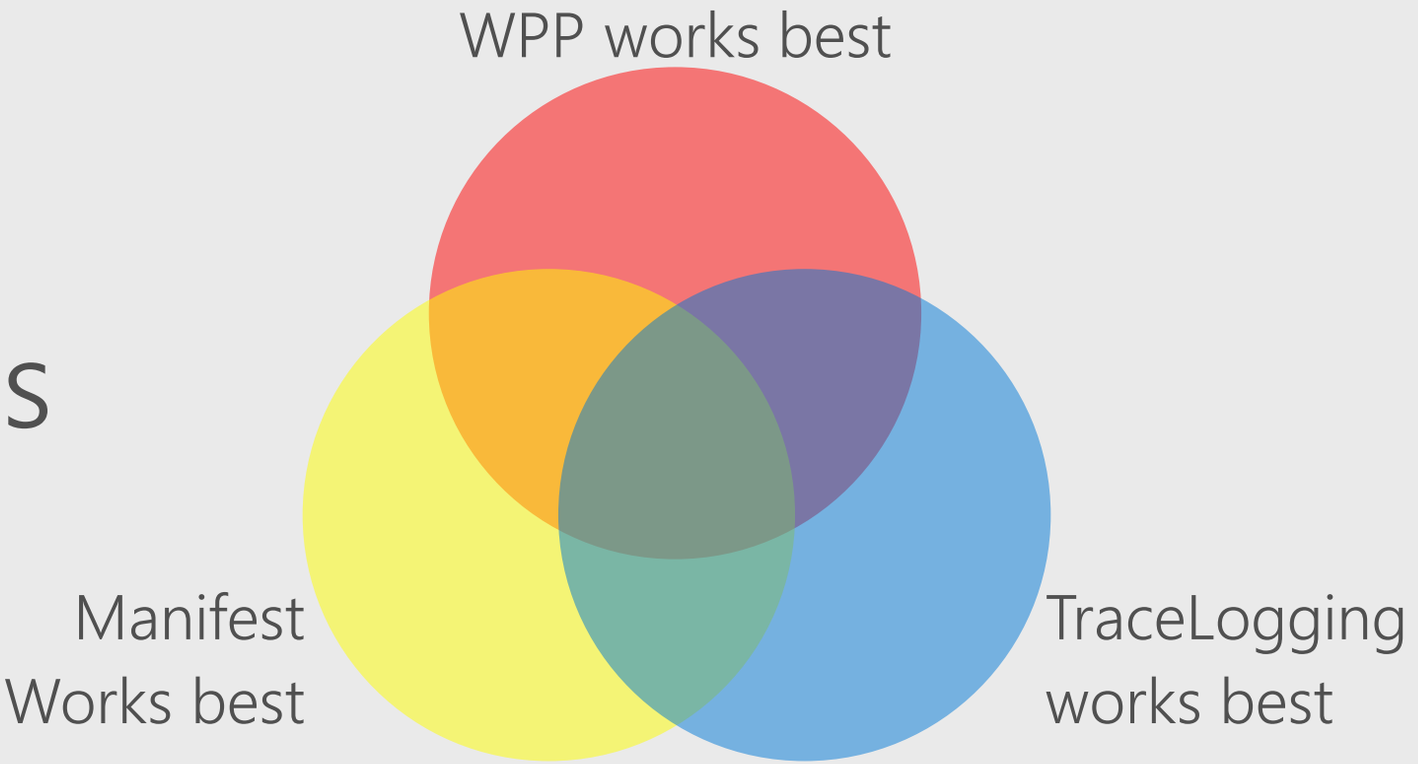
Microsoft ETW frameworks

- MOF (obsolete)
- WPP
  - Printf-style events authored in C/C++ source code (tracewpp preprocessor).
  - Decoding needs PDB (symbols) or TMF (extracted from PDB).
- Manifests
  - Structured events authored in XML manifest (mc manifest compiler).
  - Decoding needs manifest or binary manifest resources.
- TraceLogging
  - Structured events authored in source code.
  - Decoding always works (decoding information is inside the event).

Which framework is best?

- It depends.
- Each framework is best in at least one scenario.
- That's why there are 3 frameworks!

ETW Scenarios



Recommendations

- If your event needs to work with Event Log (i.e. it is of interest to a system administrator), use manifest-based ETW.
- If log size is a serious concern (i.e. high-frequency events), use manifest-based ETW (or consider WPP).
- If you need to keep the metadata private, use manifest-based ETW (or consider WPP).
- If you are happy with an existing framework, keep using it!
- For easy development and reliable decoding, use TraceLogging.

ETW is easy.

- Try out ETW.
  - C/C++: TraceLoggingProvider.h in Windows 10 SDK.
  - .NET: System.Diagnostics.Tracing.EventSource in .NET 4.6.
  - Windows Framework: Windows.Foundation.Diagnostics.LoggingChannel in Windows 10.
- Updated SDK tools in Windows 10 Creators Update.
  - Tracelog
  - Tracefmt
  - Traceview
  - xperf

Resources

- Blog: https://blogs.msdn.microsoft.com/dcook/
  - ETW Overview
  - TraceLogging Background
- Tracing tools: https://msdn.microsoft.com/en-us/windows/hardware/drivers/devtest/tools-for-software-tracing
- Re-visit Build session recordings on Channel 9.
- Continue your education at Microsoft Virtual Academy online.

#MSBuild

Microsoft