Consuming Events (Event Tracing) - Win32 apps

By Karl-Bridge-Microsoft

Archived: 2026-04-06 00:19:46 UTC

Event trace consumers can process events from one or more providers. Consumers can process events from a log file or in real time. You can consume events in real time only if the controller specifies the real time logging mode for the session. For performance reasons, real-time processing is not recommended prior to Windows Vista.

To specify the trace session from which you want to process events, you use the **EVENT_TRACE_LOGFILE** structure. You must initialize a copy of this structure for each log file or real time session that you want to process.

To consume events from a log file, set the **LogFileName** member to the name of the log file. To consume events from real time session, set the **LoggerName** member to the session name. You also use this structure to specify the *BufferCallback* callback and the *EventCallback* or **EventRecordCallback** callback used to process the events.

- **EventRecordCallback**—Receives and processes all events (including the header event) from one or more log files and a real-time session. You implement this callback if you use the trace data helper functions to parse the event data or you want to retrieve metadata about the event.
- *EventCallback*—Receives and processes all events (including the header event) from one or more log files and a real-time session.
- *BufferCallback*—Receives and processes summary information about the current buffer, such as events lost. ETW calls the callback after delivering all events in the buffer to the consumer. The consumer can also use this callback to cancel event processing; however, if you are consuming events in real time, ETW delivers events until the controller stops the session.

After defining one or more trace sessions, call the **OpenTrace** function for each trace session that you want to process; you can process events from one or more log files, but from only one real-time session. You then pass the list of trace session handles that **OpenTrace** returns to the **ProcessTrace** function. The **ProcessTrace** function combines the events, sorts them into chronological order, and then delivers them to the callbacks one at a time. The events can be filtered to include only those that fall into a specific time frame using the *StartTime* and *EndTime* parameters. The **ProcessTrace** function blocks the thread until your consumer processes all events in the trace sessions, the *BufferCallback* returns **FALSE**, or you call **CloseTrace**.

Prior to Windows Vista: You can call **CloseTrace** only after **ProcessTrace** returns.

For an example that shows how to consume events published using a manifest, MOF, or TMF files, see Retrieving Event Data Using TDH. Note that beginning with Windows Vista, you should use the trace data helper (TDH) functions to consume events.

For an example that shows how to consume events published using MOF, see Retrieving Event Data Using MOF.
