

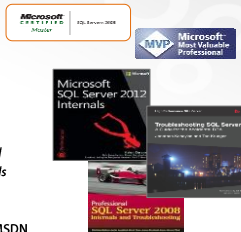
Everything You Never Wanted to Know About Extended Events

Jonathan Kehayias, Principal Consultant
SQLskills



Author/Instructor: Jonathan Kehayias

- Consultant/Trainer/Speaker/Author
- Principal Consultant, SQLskills.com
 - Email: Jonathan@SQLskills.com
 - Blog: <http://www.SQLskills.com/blogs/Jonathan>
 - Twitter: @SQLPoolBoy
- Contributing Author: *SQL Server 2008 Internals and Troubleshooting*, *Microsoft SQL Server 2012 Internals*
- Microsoft Certified Master: SQL Server 2008
- SQL Server MVP since October 2008
- Author of Using Extended Events whitepaper on MSDN
- Developed Extended Events Manager SSMS Addin for SQL Server 2008 Open Source project on Codeplex
- Member of the Tampa SQL Server User Group and PASS



2

© SQLskills All rights reserved.
<http://www.sqlskills.com>



- Team of world-renowned SQL Server experts:
 - Paul S. Randal (@PaulRandal)
 - Glenn Berry (@GlennAlanBerry)
 - Jonathan Kehayias (@SQLPoolBoy)
 - Kimberly L. Tripp (@KimberlyLTripp)
 - Erin Stellato (@ErinStellato)
 - Tim Radney (@TRadney)
- Instructor-led training: Immersion Events (US, UK, and Australia)
- Online training: pluralsight.com
- Consulting: health checks, hardware, performance, upgrades
- Remote DBA: system monitoring and troubleshooting
- Conferences: PASS Summit, SQLIntersection
- Become a SQLskills Insider
 - <https://www.sqlskills.com/insider>



3

© SQLskills All rights reserved.
<http://www.sqlskills.com>

2017 Immersion Events

- **Classes in Chicago**
 - IE0: Immersion Event for Junior/Accidental DBA - **October 2-4**
 - IEPT01/2: Immersion Events on Performance Tuning – Parts 1 and 2 - **October 2-6**
 - IEAG: Immersion Event on High Availability and Disaster Recovery - **October 5-6**
 - IEAzure: Immersion Event on Azure - **October 9-11**
- **In-depth, instructor-led, technical training for SQL Server**
- **Here's our topic list for IEPT02: Immersion Event on Performance Tuning, Part 2**
- **For more information:** <https://www.sqlskills.com/training/>

SQL Server I/O	I/O Concepts for DBAs	SANs for DBAs
SQL OS Scheduling and CPU Performance Tuning	SQL OS Memory Management and Memory Performance Tuning	Data Collection and Baselineing
Wait and Latch Statistics	Query Plan Analysis	Extended Events
Performance Issue Patterns	Statement Execution, Stored Procedures, and the Plan Cache	
Deadlock Analysis	Advanced Extended Events	

SQLskills

4

© SQLskills. All rights reserved.
<https://www.sqlskills.com>

pluralsight

- Email paul@SQLskills.com with the subject line: **SQLSaturday Pluralsight code** to get a **FREE (no catches, no credit card)** 30-day trial of our 140+ hours of SQLskills content on Pluralsight
- **For example:**
 - <http://www.pluralsight.com/courses/sqlserver-deadlocks>
 - 2.5hrs on deadlocks, analysis, prevention
 - <http://www.pluralsight.com/training/Courses/TableOfContents/sqlserver-waits>
 - 4.5 hours on waits, latches, spinlocks
 - <http://www.pluralsight.com/courses/sqlserver-optimizing-stored-procedure-performance>
 - 7 hours on stored procedure performance tuning

SQLskills

5

© SQLskills. All rights reserved.
<https://www.sqlskills.com>

Agenda

- Transitioning from Trace/Profiler to Extended Events
- Extended Events Core Concepts
- Extended Events UI
- System Health Event Session
- Event Session Options
- Advanced Targets

SQLskills

6

© SQLskills. All rights reserved.
<https://www.sqlskills.com>

History Review

- SQL Trace was introduced in SQL Server 6.5 as a graphical tool, which was external to the SQL Server architecture
 - Saved output to a trace or script
 - Very limited in what it could capture
- SQL Profiler was introduced in SQL Server 7.0, and provided new capabilities for capturing and analyzing data

Basic Uses of SQL Trace

- Provides real-time insight into SQL Server activity
- Allows DBAs to capture duration, frequency, and resource use of queries
- Can be used to capture a baseline, or replay a workload (one client)
- May be used to audit user activity
- Most frequently used to troubleshoot performance issues and errors

How SQL Trace Works₍₁₎

- There is a trace controller within the engine that maintains a bitmap of the events that are being captured by an active trace
- Event providers check to see if their event is active in that bitmap, and if so, provide a copy of the event data to the trace controller
- The trace controller queues the event data, and provides the event data to all active traces collecting the event

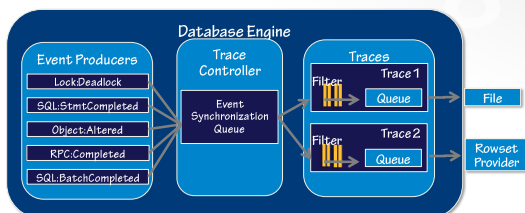
How SQL Trace Works₍₂₎

- The individual traces filter the event data and remove any columns that are not needed, and also discard events that do not match the filter
- The remaining event data is written to a file locally, or buffered to the row-set provider for consumption by external applications like SMO and SQL Profiler

How SQL Trace Works₍₃₎

- The file provider is designed to guarantee that no data will be lost
 - During I/O pressure, internal buffers will fill up if disk writes cannot keep up
 - Once those buffers fill up, the threads sending the event data to the trace will have to wait for buffer space
- The rowset provider is *not* designed to make data loss guarantees
 - If data is not consumed fast enough, the internal buffers fill and events are jettisoned after 20 seconds

SQL Trace Architecture



Now it's time to say goodbye...to Trace₍₁₎

- Extended Events is the replacement for SQL Trace
- Trace is deprecated as of SQL Server 2012
- XE provides the same functionality as Trace:
 - Capture information about what's going on inside SQL Server
 - Choose which fields to capture
 - Filter on different fields
 - Automate capture

Now it's time to say goodbye...to Trace₍₂₎

- Extended Events has a UI in SQL Server 2012+
 - No need to use XQuery to view the data (*with some exceptions*)
- Extended Events is more lightweight than Profiler and SQL Trace, and can do more than SQL Trace ever could
- XE is the only method for tracing *new* SQL Server features

What are Extended Events?₍₁₎

- Advanced event collection infrastructure introduced in SQL Server 2008 and provided by SQLOS
- Highly-flexible implementation which allows complex configurations for event collection that simplify problem identification

What are Extended Events?(2)

- Provides the ability to perform performance troubleshooting that's not possible any other way:
 - Find the accumulated effect of queries using the query_hash
 - Identify stored procedures that exceed previous max duration, CPU, or I/O values
 - Identify statement timeouts/attention events
 - Capturing the first N executions of an event
 - Find statements that recompile most frequently

Demo

- Transitioning from Profiler to Extended Events

Changes in XE by SQL Server Version

SQL Server Version	Number of Events	Notes
2008 SP3	253	
2008 R2 SP2	262	
2012 SP1	625	Includes all events available in Trace
2014	870	v. 12.0.2402
2016	1326	v. 13.0.4435
2017 CTP3	1600	

- There are only 180 Trace events in each of the versions listed here

Core Concepts: Packages

- **Packages are loaded by individual modules at run-time**
 - Modules: sqlservr.exe, sqldk.dll, sqlos.dll, sqlmin.dll, sqlang.dll, hkengine.dll
- **Packages are containers that define the available Extended Events objects and their definitions**
- **Packages are not a functional boundary of usage**
 - Objects from one package can be used with objects from another package

Core Concepts: Events

- **Correspond to well-known points in the code**
 - e.g. a T-SQL statement finished executing; a deadlock occurred
- **Deliver a basic payload of information**
 - Payload is defined by a versioned schema
 - Data elements are always returned by an event; customizable (optional) data elements are only collected when specified
- **Stored in binary, and exposed as XML through T-SQL**
- **Defined using the Event Tracing for Windows (ETW) model to allow integration with ETW**

Core Concepts: Predicates

- **Define the condition required for an event to actually fire**
 - Implemented as Boolean expressions that can be grouped into logical blocks for complex analysis
- **Support short-circuit evaluation**
 - First false evaluation of a logical block of predicates prevents the event from firing
- **Operate on global or event payload data**

Core Concepts: Predicates (2)

- Use basic arithmetic operators, or textual comparators for more complex expressions
 - e.g. greater than last max value, less than last min value, and divides evenly by int64 can perform event sampling which is not possible with basic Boolean expressions
- Predicates can store state (this is pretty cool)
 - e.g. count the number of times an event fires and only publish the event every N times
- Inverse predicates can be defined using NOT, !=, or <>

Core Concepts: Actions

- Perform additional operations when an event fires
 - e.g. collect additional state data, perform a memory dump
- Only execute after predicate evaluation determines the event will fire
- Execute synchronously on thread that fired the event
- Any action may be used with any event, but state data may not be available at the point in code where an event fires

Core Concepts: Targets

- Event consumers - single event or a buffer full of events
- Both synchronous and asynchronous targets exist
- Basic targets collect raw event data
 - event_file
 - ring_buffer
- Aggregate targets group data based on criteria
 - histogram (event bucketizer)
 - event_counter
 - pair_matching (match events)

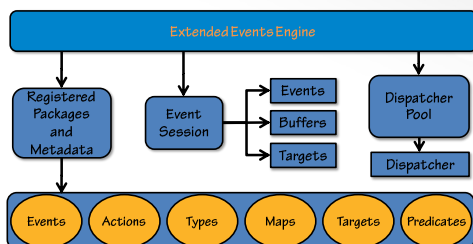
Core Concepts: Types and Maps

- **Types** define the data type for an event column, action, or global predicate source (e.g. integer, string)
- **Maps** provide a lookup to convert a value from one format to another
 - e.g. a numeric lock mode to a string describing the lock mode – it's easier to understand mode = "X" than mode = 5
- **Maps are used as types in the Extended Events Engine**
 - Defining a predicate on a map based event column requires using the map_key value from the sys.dm_xe_map_values DMV

Core Concepts: Event Sessions

- **Collection of events, their configured actions and predicates, and targets to store the event data**
- **Functional boundary for configuration**
 - Multiple sessions can have the same events with different predicates and targets
- **Per session memory buffers store data generated by events before being dispatched to targets**
- **Can track causality of event firing (linking events together) and be configured to start automatically**

Extended Events Architecture



Metadata Views

- **sys.dm_xe_packages**
 - Contains an entry for each of the packages registered in the engine
 - Each package will have a unique guid, used to map its objects to it
- **sys.dm_xe_objects**
 - Contains information about the objects (events, actions, predicates, targets, types and maps) available in the engine
 - Objects have the `package_guid` of the package that loaded them
 - The `object_type` column determines the type of object

Metadata Views

- **sys.dm_xe_object_columns**
 - Contains information about the columns, or data elements, that exist for a specific object
 - `readonly` - system metadata about an event
 - `data` - the data elements that are returned when the event fires
 - `customizable` - control collection of additional data elements in the event payload that have a higher cost to collect
- **sys.dm_xe_map_values**
 - Contains key/value pairs for each of the maps defined in the system
 - Linked by `object_package_guid` to the package that created them

Demo

- Understanding Extended Events Architecture

Management DDL₍₁₎

- **All DDL requires:**
 - ALTER ANY EVENT SESSION in 2012+
 - CONTROL SERVER in 2008/2008R2
- **CREATE EVENT SESSION**
 - Creates a new event session based on the events, actions, predicates, targets, and session options provided
 - Requires at least one event
 - All event sessions are created in a stopped state

Management DDL₍₂₎

- **ALTER EVENT SESSION**
 - Alter the state of an event session to start or stop
 - Add or remove events and/or targets from an event session
 - Change session configuration options for a stopped event session
- **DROP EVENT SESSION**
 - Removes an event session from the system entirely
 - Memory resident targets are not available after an event session is dropped

Creating an Event Session

```
CREATE EVENT SESSION [Track_Queries]
ON SERVER
ADD EVENT sqlserver.sp_statement_completed (
SET collect_statement=(1)
ACTION (
sqlserver.database_id,
sqlserver.session_id)
WHERE (logical_reads > 25000)
)
ADD TARGET package0.ring_buffer
(SET max_memory=(2048))
WITH (MAX_MEMORY=4096 KB,
MAX_DISPATCH_LATENCY=30 SECONDS);
```

Working with Event Sessions₍₁₎

```
ALTER EVENT SESSION [Track_Queries]
ON SERVER
STATE=START;
GO

ALTER EVENT SESSION [Track_Queries]
ON SERVER
ADD EVENT sqlserver.sql_statement_starting(
SET collect_statement=(1)
ACTION (
sqlserver.database_id,
sqlserver.session_id)
);
GO
```

Working with Event Sessions₍₂₎

```
ALTER EVENT SESSION [Track_Queries]
ON SERVER
DROP EVENT sqlserver.sp_statement_completed;
GO

ALTER EVENT SESSION [Track_Queries]
ON SERVER
DROP TARGET package0.ring_buffer;
GO

ALTER EVENT SESSION [Track_Queries]
ON SERVER
STATE=STOP;
GO
```

Demo

- Altering event sessions

Dispatching

- The dispatching of events to the asynchronous targets is handled by the dispatcher pool in the Extended Events Engine (in SQLOS)
- Dispatch occurs under two conditions:
 - The memory buffer becomes full
 - The event data in the buffer exceeds the event session's MAX_DISPATCH_LATENCY configuration option

Actions that Add Data

- Actions execute synchronously on the firing thread and should be used only where they actually add benefit to the event data
- Don't add actions to do event correlation, use causality tracking instead
- Actions cause side effects to occur when an event fires
 - Performing a memory dump of a single thread
 - Performing a memory dump of all threads
 - Inserting a debug break (this really does what it says it does!)

Ring Buffer Target₍₁₎

- General purpose, in-memory target that does FIFO event collection
- Ideal target for small data sets where event loss is acceptable
- Data is in binary format, but materialized as XML when the target is queried via the sys.dm_xe_session_targets DMV
 - If viewed using the UI, the XML is shown, not the processed events

Ring Buffer Target₍₂₎

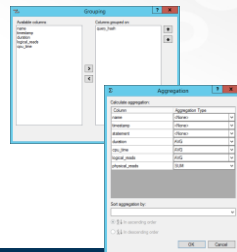
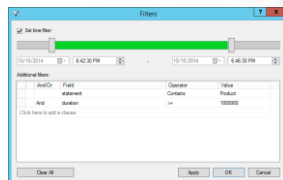
- **Options:**
 - MAX_MEMORY: the maximum amount of memory in KB to be used, older events are dropped when this limit is reached
 - MAX_EVENT_LIMIT: the maximum number of events to keep, existing events are dropped when this limit is reached
 - OCCURRENCE_NUMBER: number of each event type to keep
- **Data is lost when the event session is stopped**
- **DMV limitations may result in unreadable XML from sys.dm_xe_session_targets**

Event File Target₍₁₎

- **Similar to the trace file in SQL Trace, this target collects event data in a file on disk**
 - Data persists after an event session is stopped
 - Files can be shared
 - Useful for long-term and/or detailed analysis
- **Event data has the same XML schema as the ring_buffer target**

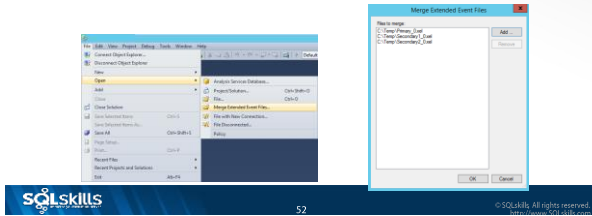
Event File Target₍₂₎

- **Can be read using the UI in SQL Server 2012 and higher**
- **In 2008 and 2008 R2:**
 - Target data must be queried through the SQL Server database engine, there is no external API available for reading the file
 - But you can view the files in the 2012 UI!
 - File target uses two files, log and metadata, and both must exist to read the data
 - The metadata file simply describes the structure of the log file, not the events themselves

[illegible]

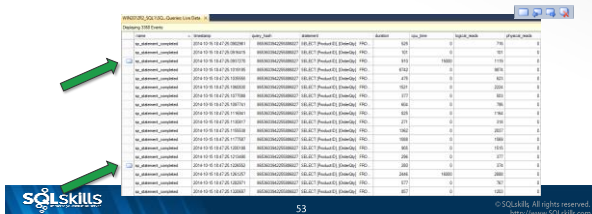
Data Viewer: Merging Files

- Multiple files can be merged together in the UI to create one master file for analysis



Data Viewer: Bookmarking Events

- Within the UI, bookmark events to quickly find them again



Demo

- Eliminating ClearTrace

system_health Background

- First implemented in SQL Server 2008 and is running by default on all installations from SQL Server 2008 onwards
- Implemented in conjunction with the SQL Server Product Support group to collect system data that you can use to help troubleshoot performance issues in the Database Engine
- This session starts automatically when the SQL Server Database Engine starts, and runs without any noticeable performance effects under most circumstances

system_health in SQL Sever 2012+

- **Addition of an event_file target**
 - Maintains four rollover files 5MB each
 - Ensures data collected is available in the event of an instance failure or restart
- **New events added for better diagnostic of problems after they occur**

Discovering events being collected

- XML deadlock reports
- High severity errors (above severity 20) and OOM errors
 - 17803, 701, 802, 8645, 8651, 8657, 8902
- CLR allocation failures
- SQLOS ring buffer recorded events
 - Scheduler, connectivity, memory broker/node, security error
- Waits
 - Latches and important non-lock resources exceeding 15 seconds
 - Locks that have exceeded 30 seconds
 - Login related preemptive waits exceeding 5 seconds others exceed 45 seconds
- sp_server_diagnostic component output

Demo

- Discovering events being collected

Session Definition Catalog Views

- **sys.server_event_sessions**
 - Contains the name and session level options for each event session that exist inside of the Extended Events Engine
- **sys.server_event_session_events**
 - Contains information about the events and predicates that are a part of an event session
- **sys.server_event_session_actions**
 - Contains one row for each action for each event in an event session

Session Definition Catalog Views

- **sys.server_event_session_targets**
 - Contains one row for each of the configured targets that are defined for an event session
- **sys.server_event_session_fields**
 - Contains one row for each of the configured options for each target defined for an event session

Demo

- Querying Catalog Views

Active Session DMVs

- **sys.dm_xe_sessions**
 - Contains the event session name, memory buffer configuration, event loss information, and date/time the event session was started for each active event session (STATE=START) in the SQL Server Instance
- **sys.dm_xe_session_events**
 - Contains information about each of the events and the event predicate for events defined in an active event session
- **sys.dm_xe_session_event_actions**
 - Contains one row for each action that is defined on an event in an active event session

Active Session DMVs

- **sys.dm_xe_session_targets**
 - Contains the name, type of target, execution statistics and an XML column for each target that exists for an active event session
 - The data for memory resident targets is in the xml column and for persisted targets this will contain execution statistics
- **sys.dm_xe_session_object_columns**
 - Contains information about the configuration options of each of the targets, as well as information about the customizable columns for events in the event session

Demo

- Querying Active Session DMVs

Events of interest in SQL Server 2012/2014

- `xml_deadlock_report` – historical deadlock information collected by default (no more trace flags)
- `*_ring_buffer_recorded` – historical information about SQLOS ring buffer entries persisted beyond in memory/after instance restarts
- `sp_server_diagnostics` – instance health information in 5 minute intervals (our new window to the past)
 - System – dumps, spinlocks, latch warnings, CPU usage, etc
 - Resource – memory status and performance counters
 - Query Processing – wait info, blocking, CPU intensive requests, etc
 - IO Subsystem – long or pending I/O requests

Demo

- Viewing `system_health` history in the UI

A brief introduction to XML in SQL

- **XML methods allow data access in the XML data type**
 - `.query()` – returns a XML document from the XPATH expression
 - `.value()` – returns a scalar value from the XPATH expression
 - `.nodes()` – shreds XML into relational data, each result from the XPATH expression becomes a new XML document in a row
 - Most commonly as a FROM/CROSS APPLY against the XML source
- **XPATH expressions generally have the form of:**
 - `/node/node`
 - `/node/node[@attribute]`

Demo

- Querying `system_health` history with Transact-SQL and XQuery

Minimizing Event Session Overhead

- **Predicate definition order matters**
 - Remember that short-circuit logic is applied to predicate evaluation
 - First logical block to evaluate to false prevents event firing
 - Filter on event data elements before global predicate fields whenever possible to prevent collection of the global field
- **Predicates that maintain state reduce event firing frequency but must be the last predicate**
 - `package0.counter` to limit event execution to N times
 - `package0.greater_than_max_uint64` for duration limitations

Demo

- Defining Predicates

Event Counter Target

- Counts how many events occurred for the event session
- Use when defining an event session for an unknown workload to determine how many events you can expect based on your event sessions definition as a part of planning the appropriate targets to use for the session

Demo

- Event Counter Target

Managing Memory Buffer Size

- Memory buffers for an event session are configured based on the `MEMORY_PARTITION_MODE` and the `MAX_MEMORY` session level options
 - Default `MEMORY_PARTITION_MODE=NONE` and `MAX_MEMORY=4MB` – result is 3 buffers that are 1.4MB
- `MAX_MEMORY` is not the actual max memory for the event session since buffers align on ~64K boundaries
 - Minimum buffer size is ~128KB or the event session won't start

Effect of `MEMORY_PARTITION_MODE`

- `MEMORY_PARTITION_MODE` determines the number of memory buffers created for the event session
 - `NONE=3` buffers
 - `PER_NODE=3` buffers per NUMA node
 - `PER_CPU=2.5` buffers per scheduler (even OFFLINE schedulers)
- Partitioning on large NUMA systems `PER_NODE` will allow schedulers local access to memory buffers for events
- `PER_CPU` partitioning is best for large workloads

Demo

- Memory Buffer Management

Options to Control Event Loss

- **EVENT_RETENTION_MODE**
 - Determines whether single events, entire buffers, or no events can be lost by the event session
 - No event loss can significantly impact performance
- **MAX_EVENT_SIZE**
 - Determine what the maximum event size may be based on the event definitions
 - Can not be smaller than a single memory buffer for the event session and the large buffers will be in addition to the regular buffers for the session

Other Event Session Options

- **MAX_DISPATCH_LATENCY**
 - Dispatch events to targets faster
 - May impact the dispatcher pool if multiple event sessions exist
- **TRACK_CAUSALITY**
 - Attaches a GUID and sequence number to events for correlation of what events led to other events
 - E.g. watching all the page splits that occur from a single insert/update
- **STARTUP_STATE**
 - Start the event session automatically with SQL Server

Demo

- Event Session Options

histogram Target

- Prior to SQL Server 2012, this was implemented as two separate targets: `synchronous_bucketizer` and `asynchronous_bucketizer`
- Aggregates events into buckets creating a histogram (hence the name) of the frequency that events fire
- Does not store actual event data, only the count of occurrences – no additional aggregations are possible
 - This level of analysis requires collecting the data to either the `ring_buffer` or `event_file` and post processing of the events

histogram Target Options

- **slots: the number of buckets to maintain**
 - The value specified is rounded up to the next power of 2 with a default of 256
- **filtering_event_name: any event that in the session**
 - This option is optional, but specifies the specific event to bucket on in the format of `package_name.event_name`
- **source_type: specifies the type of object to bucket on**
 - This option has two values:
 - 0 = an event in the session
 - 1 = an action in the session (the default)
- **source: the event column or action to bucket on**

Identifying Recompilations

- Multiple events track recompile occurrences:
 - `sqlserver.sql_statement_starting/sqlserver.sp_statement_starting` contain the state column to determine if a recompile occurred
 - SQL Server 2012+ `sqlserver.sql_statement_recompile` provides more details about the recompilation including the exact cause
- Using performance counters it is impossible to determine which database has the highest recompile issues

Finding Root Cause of Recompilation

- Using performance counters it is impossible to determine which database has the highest recompile issues
- Capture data using histogram and bucket on `sqlserver.database_id` in 2008/R2 or `source_database_id` in 2012 to find the worst database
- Filter by `database_id` and change histogram to bucket on `sqlserver.tsql_stack` in 2008 or on `statement/object_id` in 2012 to find recompiling statements

Demo

- Histogram Target

pair_matching Target

- Specialized target that matches events based on a specified criteria and discards the matched pairs, leaving unmatched events only in the data
- Careful selection of the pairing criteria has to be made or invalid event pairings will occur resulting in incorrect analysis
 - E.g. `lock_acquired` and `lock_released` are not fired the same number of times when lock escalation occurs
- Provides similar XML output as the `ring_buffer`

pair_matching Target Options

- **begin_event:** any event that in the session
 - This option specifies the beginning event to be paired
- **end_event:** any event that is in the session
 - This option specifies the ending event to be paired
- **begin_matching_columns:** comma-delimited and ordered list of column names for matching events on
- **end_matching_columns:** comma-delimited and ordered list of column names for matching events on

pair_matching Target Options (2)

- **begin_matching_actions:** comma-delimited and ordered list of action names for matching events on
- **end_matching_actions:** comma-delimited and ordered list of action names for matching events on
- **respond_to_memory_pressure:** specifies the target response to memory pressure
 - Can be set to two values:
 - 0 = Do not respond to memory pressure (the default)
 - 1 = Respond to memory pressure by not adding new orphans

pair_matching Target Options (3)

- **max_orphans:** specifies the maximum number of unpaired events that will be collected in the target
 - Once the limit is reached, unpaired events are removed on a first-in, first-out (FIFO) basis
 - The default for this option is 10,000

Troubleshooting Invalid Pairs

- Troubleshooting incorrect matching should be done using `TRACK_CAUSALITY` and adding a `ring_buffer` or `event_file` target to the event session
 - When an unmatched event occurs the `attach_activity_id` guid can be used to lookup all of the events in the `ring_buffer` or `event_file` target data to determine why the event was unmatched
- Ex: unmatched event error is the `statement_starting` and `statement_completed` events when a recompile occurs
 - In this scenario the `statement_starting` event actually fires twice, once for the recompile and once for the actual execution

Execution Timeouts

- Default .NET execution timeout is 30 seconds
- The statement starting and completed events should both fire under normal query execution, but during a client timeout, the completed event will not be fired
- Pairing the events with the `pair_matching` target on the `sqlserver.session_id` and `sqlserver.tsq_stack` actions will simplify identification of timeout occurrences
 - The starting event for the timeout will not pair up to a completed event and will remain in the target when the timeout occurs

Demo

- Tracking Statement Timeouts

Extended Events Object Model

- The XEStore object provides access to two sources:
 - Metadata – includes the packages and associated metadata for the Extended Events objects available in the instance
 - Runtime – the event sessions that are defined on the instance
- The intention behind the object model is to support the programmatic creation of sessions from SSMS only
 - Creating sessions from Transact-SQL DDL commands is often faster and requires less code than creating them programmatically inside of .NET

Reading Files with QueryableXEventData

- The QueryableXEventData object provides multiple methods of reading file data from Extended Events
 - Single string to a single path
 - Can be an explicit file path or a wild card string to process all files matching the wild-card expression
 - Single string array containing a list of multiple XEL files to process
 - Two string arrays containing a list of multiple XEL files to process along with a list of multiple XEM files to process for the XEL files
 - Provides backwards compatibility to SQL Server 2008 and 2008 R2 files

Reading Live Data with QueryableXEventData

- The live event_stream target can only be connected through the use of the .NET API, it can't be configured for use through standard DDL statements
- To read a live stream with QueryableXEventData requires:
 - connectionString – the connection string to connect to the server
 - source – the event session to attach the stream to
 - EventStreamSourceOptions – the source of the event stream
 - EventStreamCacheOptions – specifies the cache options of the event stream

event_stream Target

- Can only be leveraged through the .NET API – No DDL
- Implements shadow copies of the event session buffers
- Changes the MAX_DISPATCH_LATENCY to 3 seconds for the event session to provide a "live" view of the data
 - Stream is asynchronously served by the session
 - The MAX_DISPATCH_LATENCY will automatically revert back to the session-configured value when the stream is closed
 - Failure to consume the events efficiently will result in the streaming provider being closed by the session to prevent performance impact to SQL Server

Considerations for Using the .NET API

- Necessary assemblies are not a part of SQL Management Objects (SMO) or SQL Server Feature Pack downloads
 - Creates licensing challenges for redeploying applications that depend on the assemblies required
- Prior to SQL Server 2012 Service Pack 1 only the x86 versions of the assemblies were shipped
 - Requires referencing the following assemblies
 - Microsoft.SqlServer.Management.XEvent
 - Microsoft.SqlServer.Management.XEventEnum
 - Microsoft.SqlServer.Management.Sdk.Sfc

Using the PowerShell Provider

- PowerShell provider is implemented as a PSDrive to manage event sessions inside SQL Server
 - SQLSERVER: drive provides access through the XEvent directory
 - From Object Explorer in SQL Server Management Studio, Start PowerShell on the Extended Events folder
 - Full path for an instance - SQLSERVER:\XEvent\ServerName\InstanceName
- Event sessions can be accessed from the Sessions folder
- New sessions can be created using the same objects provided by the .NET API for Extended Events
 - Creating sessions from Transact-SQL DDL commands is often faster and requires less code than creating them programmatically inside of Powershell