solarwinds

IMPROVE SQL SERVER PERFORMANCE MANAGEMENT WITH EXTENDED EVENTS

A Guide to Getting Started

By Kathy Gibbs, Product Manager and Janis Griffin,
Database Performance Evangelist and Senior DBA

SOLARWINDS
WHITEPAPER

INTRODUCTION

Introduced in Microsoft SQL Server 2008, Extended Events are a lightweight event-handling mechanism you can use to capture event information about the inner workings of SQL Server. Extended Events replace SQL Trace as the interface for diagnostic tracing in SQL Server 2012 and later.

You can use Extended Events as an addition to monitoring and analysis solutions like SolarWinds Database Performance Analyzer (DPA) to gain even more detailed insight into SQL Server and improve your ability to tune queries and isolate errors, including all of the following example uses:

- Collect information about wait types and the resources causing those waits
- Diagnose deadlocks, reviewing the generated XML deadlock graph and pinpointing blocking or deadlocking queries
- Understand why page splits are happening and which T-SQL statement is causing them
- Research errors by examining the actual client-side errors as they occur
- Monitor SQL Server memory overloading
- Discover missing column statistics

Because Extended Events still require setup, must be enabled and disabled depending on what and when you need to collect, and create a large volume of raw data that must be analyzed, they are best used as an extra tool alongside a continuous, lightweight monitoring solution.

Extended Events will replace SQL Trace (or SQL Profiler with Trace) after SQL Server 2012, so it's important to become familiar with how they work, and how they can help you do your job. If you have tools and processes that depend on Trace, you'll need to find an alternate method for releases after 2012. This white paper provides an overview of Extended Events: how they're best used, how they're different from using SQL Server Profiler or SQL Trace; the Extended Events architecture; and guidelines for getting started in using them.

WHY EXTENDED EVENTS ARE BETTER THAN TRACE

Extended Events represent an evolution in Microsoft's approach to providing the in-depth, technical information about the workings of the SQL Server database and its instances that DBAs can use for in-depth analysis of complex database operations and errors.

Until the introduction of Extended Events, this low-level information was primarily available only through use of SQL Trace, or SQL Profiler with Trace. Trace, however, has significant limitations. It uses system resources to gather data, and can incur a significant performance cost. If, for example, you're using Trace to troubleshoot a performance problem, the act of using Trace can make that problem even worse. Trace does not produce historical or trending information, and

the snapshot view produced can skew or hide real underlying issues. Finally, Trace produces a very large amount of output, and it can be challenging to accurately assess the meaning of the information collected. Tools that depend on Trace to capture raw data and then parse it to provide real time monitoring information are bound by the large output and heavy loads imparted by the Trace.

By contrast, Extended Events are very lightweight; according to one estimate, Microsoft predicted that 20,000 events/sec on a 2 GHZ Pentium with 1 GB RAM would use less than 2% of the CPU. (Of course then you'd need to figure out how to analyze 20,000 events per second!) Extended Events create event data survivability, and so can be used to generate large numbers of events that are then accessible for historical and trend analysis. And, Extended Events provide an extremely powerful and flexible architecture for configuring as much or as little information collection as is necessary to troubleshoot a problem. Additionally, Extended Events enable you to correlate database performance information with event tracing for Windows. With Extended Events, you can get precisely the information you need, and nothing more, at just the time you need it.

WHAT EXTENDED EVENTS CAN AND CAN'T DO

While all these benefits are a significant improvement over using Trace, there remain a few disadvantages to the tools SQL Server provides for collecting, assessing and monitoring events over time. First, there is currently no built-in alerting. With Extended Events, information is sent to a target, where it stays until you do something with it.

Another challenge with Extended Events is that the event information collected is very low-level, detailed and high-volume. Like Trace, it is not advisable to continuously capture Extended Events. Instead, specific events should be enabled at specific times, once a problem has been identified that needs deep investigation. Because of the complexity of the raw event data, very experienced DBAs and database developers are the most likely to benefit most from the information provided; IT managers and others interested in database health and performance will need assistance in interpreting the information.

Finally, while Extended Events provide details beyond what was formerly available via Trace, it is important to note that Extended Events are still just a source of raw data that needs to be collected and analyzed to identify a problem and a solution. Extended Events provides an interface to collect a large volume of raw data, which then must still be analyzed with another tool to make the data usable. Also, Extended Events are not useful in pinpointing which issues are significant. You have to know where and when to look, just like with Trace. In fact, Extended Events are best when used with a continuous database performance monitoring solution that can help you quickly identify, prioritize and communicate issues worth investigating, such as DPA. DPA is a 24x7 performance monitoring solution that is independent of both Trace and Extended Events, and covers all servers continuously, allowing you to quickly pinpoint performance issues and drill down to the root cause and identify any need for deeper research.

IS TRACE GOING AWAY? HOW CAN I TRANSITION TO EXTENDED EVENTS?

In SQL Server 2012, the underlying mechanism for SQL Trace has been replaced with Extended Events, and SQL Trace and SQL Server Profiler will be deprecated in versions after SQL Server 2012. Tools that depend on Trace will have to be significantly overhauled or become obsolete.

The best way to transition from Trace to Extended Events is to start by understanding how Trace is currently being used. You can use the following query to discover uses of Trace in your environment:

```
SELECT instance_name, cntr_value
FROM sys.dm_os_performance_counters
WHERE object_name like '%deprecate%'
and instance_name like '%trace%
```

Once you've identified the uses of TRACE, you can begin to work with the application owners to determine how that collected information is really being used, and then begin planning for an alternative solution. This might be a good time to consider, for example, using a lightweight, continuous performance monitoring solution such as DPA to better meet the needs of the application owner.

HOW EXTENDED EVENTS WORK

To understand how powerful and flexible the Extended Events system is, it's helpful to understand the Extended Events architecture. While **Microsoft Books Online provides a very detailed explanation of the architecture**, we'll provide a high-level overview here for convenience.

At the core is the Extended Events Engine, which implements and manages an event session. The Engine allows you to capture events (such as wait events, or locks, or errors) and specific information about those events.

To collect event data in **Extended Events**, you must create and configure a session that specifies exactly what data to collect. A session identifies a source of the events, the target to which event information is sent, and the actions or filters (predicates) to apply to the event.

A **module** is a container for packages, and is either a DLL or EXE.

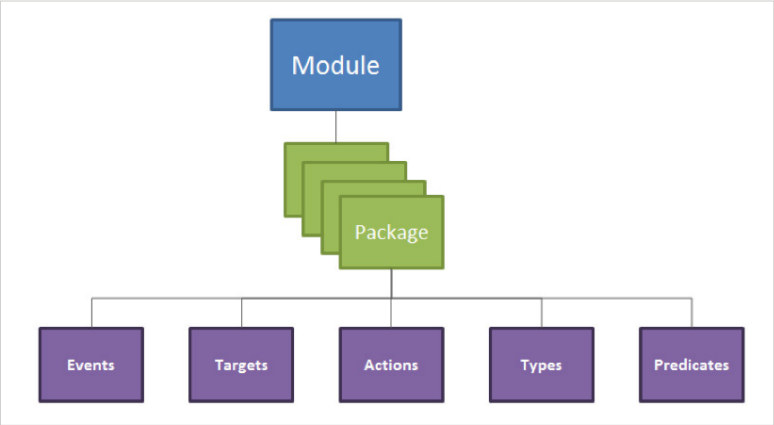


Figure 1: *Extended Events Modules*

Packages are containers for events, and are identified by a GUID. There are four packages provided with SQL Server:

- sqlos
- sqlserver
- package0
- secaudit – (use of this package is reserved for the SQL Server audit facility)

Packages contain Extended Events objects, and all the packages contain between them all the Extended Events object types. A package does not have to contain all object types, and may use objects from other packages.

Object types include:

- **Events**—specifies which trace events are to be monitored.
- **Targets**—also known as consumers, targets receive the collected event information. Targets are both event- and consumer-agnostic, so any event can be processed by any consumer. Separating the target from the event and the action is one of the features that make Extended Events such a flexible tool. Some targets are synchronous and some are asynchronous, depending on their intended use.
- **Actions**—defines what should occur when an event is triggered, typically to add data or context to the event itself; for example, if you're investigating a deadlock, an action you might specify is the collection of the database names and client applications involved in the deadlock.
- **Types**—describes the format of the data collected; for example, if you're collecting data about a deadlock, you may want to capture the data as an XML deadlock report so you can graphically display it.
- **Predicates**—Boolean expressions that allow you to further filter the event data being collected, much like the WHERE clause in a query; this is what allows you to collect just the information you need and nothing more.

WHAT TOOLS DO I USE TO WORK WITH EXTENDED EVENTS?

Depending on the version of SQL Server, you use either T-SQL or SQL Server Management Studio (SSMS) to: x

- Create/modify events
- Create/start/stop session
- Manage/view session

For SQL Server 2008

For SQL Server 2008, you use T-SQL to issue data definition language (DDL) statements.

CREATE EVENT SESSION	Creates an extended event session object, Identifies Source of the events, Targets, and Parameters
ALTER EVENT SESSION	Starts/stops an event session or changes an event session configuration
DROP EVENT SESSION	Drops an event session

You can use the dynamic management views (DMVs) and catalog views added in SQL Server 2008 for this purpose to see the session data while it's being collected, as well as metadata about the event session (see Appendices A and B for listings of these views).

You can find more detailed information about **working with Extended Events in SQL Server 2008 in Microsoft Books Online.**

For SQL Server 2012

For SQL Server 2012, you use SSMS (note that you can still use T-SQL if you really want to, but we recommend using SSMS as it is much simpler).

To create or manage a session, you can work with SSMS and use the New Session Wizard as shown in the screen captures below (or bypass the wizard and use the New Session menu option):

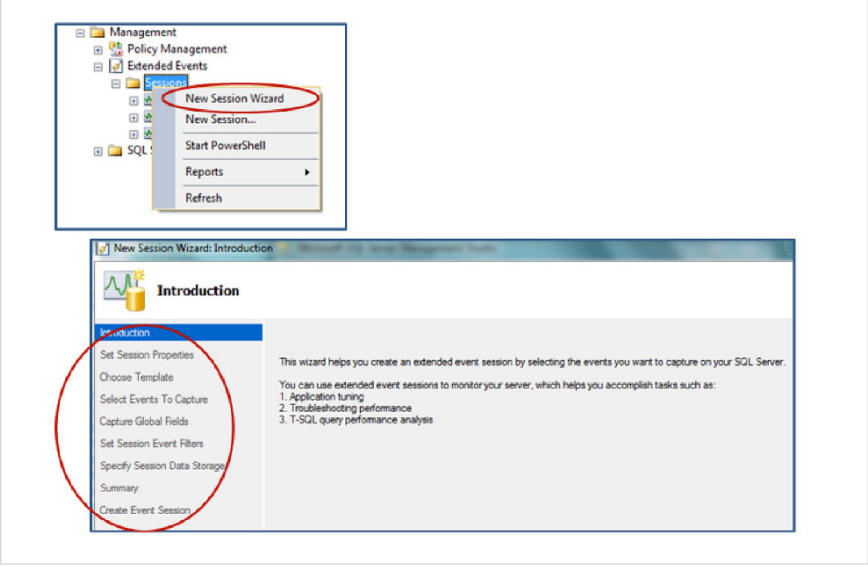


Figure 3: *Using the New Session Wizard to Create a New Extended Events Session in SSMS*

The SSMS interface for Extended Events is well developed and very easy to use. You can find detailed information about it at **Microsoft Books Online.**

HOW DO I GET STARTED?

Start by becoming familiar with the provided system_health session

To become familiar with Extended Events, we recommend that you start by examining the system_health session that SQL Server already provides.

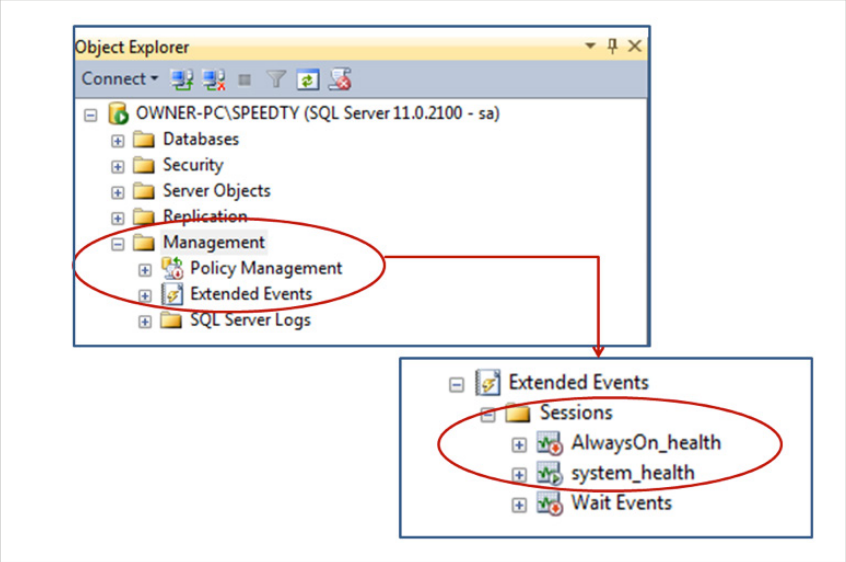


Figure 4: *Viewing the Provided system_health and AlwaysOn_health Sessions in SSMS*

This system_health session:

- Is turned on by default.
- Collects system data that you can use to help troubleshoot performance issues in the Database Engine.
- Starts automatically whenever the SQL Server Database Engine starts.
- Runs without noticeable impact on performance.

Because of these considerations, we highly recommend that you always leave this session running.

This session collects a great deal of useful information about the SQL Server database and its instances. Look at the packages the session calls, and look at the targets to see all the information it collects. Some or all of the information you might need may already be actively collected by this session, including:

- Errors with severity > 20
- Waits on latches > 15 seconds
- Waits on locks > 30 seconds
- Deadlocks
- Memory errors (for example, error 701 or 17803)
- Long preemptive or external waits

Refer to **Microsoft Books Online** for details about the information collected.

If you're using AlwaysOn, also look at the AlwaysOn_Health session

Provided for use with the AlwaysOn feature added in SQL Server 2012, this session comes with SQL Server 2012, and is turned off by default. You must specifically turn it on to see the subset of AlwaysOn related events it collects. This session is created automatically if you create an availability group, and it can be helpful if you need to troubleshoot the availability group.

SPECIAL CONSIDERATIONS WHEN WORKING WITH EXTENDED EVENTS

Resist the Temptation to Collect It All!

Extended Events give you a very robust way to collect information, and when you're first getting started, it can be very tempting to want to collect all the information you possibly can. However, this creates high overhead that you don't need and can impact performance; it also creates an overload of information that you'll have to sort through and analyze. The elegance of Extended Events, especially when compared to Trace, is that you can selectively collect just the information needed, so it's critical to become more discerning than you might have been with Trace.

Carefully Consider How You Are Using the Filters

Actions and filters can be applied to all events in a session, or just to a single event. For example, if you use the New Session wizard in SSMS (SQL Server 2012), the actions or predicates you add will apply globally to all events in the session. It's important to understand that actions and filters add to the overhead of event processing, so you will want to carefully consider how and when you apply them. For SQL Server 2012, SSMS provides useful features for controlling when predicates are applied to help you prevent potential logic issues, and you should become familiar with these as well.

Understand synchronous vs. asynchronous targets

Targets, sometimes referred to as event consumers, can write to a file, store event data in a memory buffer, or aggregate event data. Targets can process data synchronously or asynchronously, and each type has advantages and disadvantages.

Event counter	Counts all specified events that occur during an Extended Events session. Use to obtain information about workload characteristics without adding the overhead of full event collection.	Synchronous
Event file	Use to write event session output from complete memory buffers to disk.	Asynchronous
Event pairing	Use to determine when a specified paired event does not occur in a matched set (like lock acquires or lock releases)	Asynchronous
Event Tracing for Windows (ETW)	Use to correlate SQL Server events with Windows operating system or application event data.	Synchronous
Histogram	Use to count the number of times that a specified event occurs, based on a specified event column or action.	Asynchronous
Ring buffer	Use to hold the event data in memory on a first-in first-out (FIFO) basis, or on a per-event FIFO basis.	Asynchronous

If using synchronous targets, the code must wait for the event to be created and then consumed by the target; if the event is complex with many actions and filters, the code may have to wait a long time, and this will impact performance.

If using asynchronous targets, however, the buffer size can become an issue. The buffer size is controlled by the MAX_MEMORY event session option, and it is a first-in-first-out (FIFO) queue, which may fill quickly, depending on your system. When it fills, data will be lost. You can configure the EVENT_RETENTION_MODE setting to specify how an event session handles event loss if events are generating faster than they can be consumed by the target.

EXTEND YOUR DATABASE PERFORMANCE MANAGEMENT WITH EXTENDED EVENTS

Extended Events, a replacement for Trace, can provide you with a useful and very customizable framework for managing very technical, low-level event information about a SQL Server database and its instances. You can start by working with the event sessions that come with SQL Server. You can then begin creating your own sessions to aid troubleshooting in your environment, specifying just the event information you need. You can collect historical information. You can correlate the SQL Server information with Windows events. And, when paired with a tool such as DPA, which can help you quickly pinpoint the specific problems that warrant investigation, Extended Events augment your ability to provide in-depth analysis of performance issues that impact end-users.

APPENDIX A: CATALOG VIEW FOR EVENT SESSIONS

These catalog view provide useful metadata for reviewing Extended Event sessions:

sys.server_event_sessions	Lists all event session definitions
sys.server_event_session_events	Returns a row for each event in an event session
sys.server_event_session_actions	Returns a row for each action on each event of an event session
sys.server_event_session_fields	Returns a row for each customizable column explicitly set on events and targets
sys.server_event_session_targets	Returns a row for each event target for an event session

APPENDIX B: DYNAMIC MANAGEMENT VIEWS (DMVS) FOR EXTENDED EVENTS

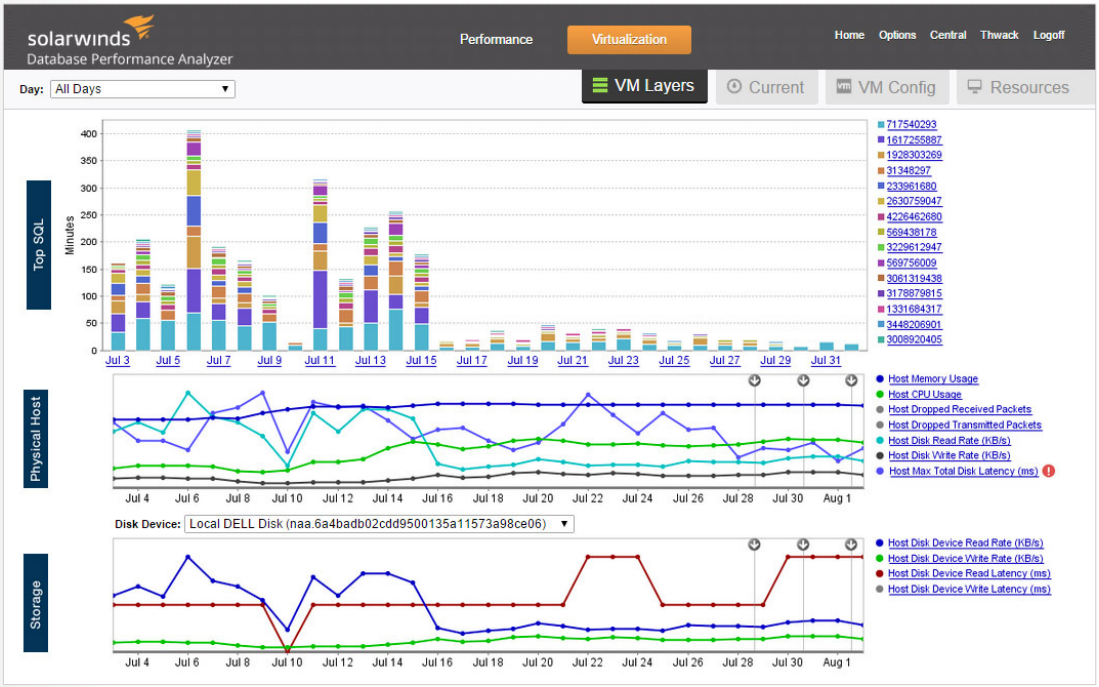
These dynamic management views (DMVs) contain session data that is created when an Extended Event session is started.

Note: These DMVs will not have session data until a session is started.

sys.dm_os_dispatcher_pools	Returns information about session dispatcher pools
sys.dm_xe_objects	Returns a row for each object exposed by an event package
sys.dm_xe_object_columns	Returns the schema information for all the objects
sys.dm_xe_packages	Lists all the packages registered with extended events engine
sys.dm_xe_sessions	Returns information about an active extended events session
sys.dm_xe_session_targets	Returns information about session targets
sys.dm_xe_session_events	Returns information about session events
sys.dm_xe_session_event_actions	Returns information about event session actions
sys.dm_xe_session_object_columns	Shows the configuration values for objects bound to a session
sys.dm_xe_map_values	Provides a mapping of internal keys to human-readable text

HOW CAN DATABASE PERFORMANCE ANALYZER HELP?

Database Performance Analyzer (DPA) from SolarWinds (NYSE: SWI) provides the fastest way to identify and resolve database performance issues. DPA is part of the SolarWinds family of powerful and affordable IT solutions that eliminate the complexity in IT management software. DPA's unique Multi-dimensional Database Performance Analysis enables you to quickly get to the root of database problems that impact application performance with continuous monitoring of SQL Server, Oracle, SAP ASE and DB2 databases on physical, Cloud-based and VMware servers.



For additional information, please contact SolarWinds at 866.530.8100 or e-mail **sales@solarwinds.com.**