|  |
| --- |
| [**Another Query to Help Find the Right Event**](http://www.sqlservercentral.com/blogs/sqlrnnr/2016/07/21/another-query-to-help-find-the-right-event/)  Posted on 21 July 2016  FROM: <http://www.sqlservercentral.com/blogs/sqlrnnr/2016/07/21/another-query-to-help-find-the-right-event/#comments>  Finding the right event or combination of events to monitor may seem like quite a daunting task with so many events to explore and (frequently) too little familiarity with Extended Events. In this follow-up article to [Extended Event Help Queries](http://bit.ly/2arQWit), I will share another means to quickly explore the Extended Events metadata in the effort to find the precise event to fit your needs.  Today I will be sharing a new query or two that I have used on more than one occasion to help track down the event(s) that I wanted to, at least, try while troubleshooting specific problems within the SQL Server Instance. I found these queries useful once again in a recent opportunity to help out some friends from the SQL Community. (Maybe, just maybe, these are really just a spin of other previously shared queries.) I will share the most recent experience in another follow-up post.  For the sake of posterity, I am also adding this to the [60 Days of Extended Events Table of Contents](http://bit.ly/XE60Days).  **Is There an Event that Contains pertinent Data?**  In my previous article I demonstrated how to find an event based solely on the name or description of the event. This is fantastic if the event name (or description) contains one of the magical words you have used. What if the event name or description has nothing to do with the terms you selected? Or, what if the data you seek may be attached to the event but wouldn’t necessarily stand out as a description for that event (by name or description details for that event)?  Now comes the more difficult task right? If the name or description of the event doesn’t relate to the search terms then you just might overlook a few events and be stuck trying to troubleshoot a problem. An equally big problem this could cause is yet another invisible barrier to using Extended Events. It would be easy to slide down the slippery slope and not transition to Extended Events just because an event, applicable to the problem at hand, could not be found.  This is where searching on the payload comes in handy. Remember from a [previous article](http://bit.ly/1K1F2Yg) that the payload is the event data that is attached to each event. Imagine how much more accurate we can become by extending our search from the name and description of the event object to next search on the payload names and descriptions? That is exactly what I will now show.  DECLARE @TermDescription VARCHAR(64) = 'Select'  , @ColumnDesc VARCHAR(64) = 'statement'  , @ReadFlag VARCHAR(64) = NULL; --readonly' --ALL if all columntypes are desired --data --customizable    SELECT oc.object\_name AS EventName  , oc.name AS column\_name  , oc.type\_name  , oc.column\_type AS column\_type  , oc.column\_value AS column\_value  , oc.description AS column\_description  , ca.map\_value AS SearchKeyword  , oc.column\_type  FROM sys.dm\_xe\_object\_columns oc  CROSS APPLY ( SELECT mv.map\_value  FROM sys.dm\_xe\_object\_columns occ  INNER JOIN sys.dm\_xe\_map\_values mv  ON occ.type\_name = mv.name  AND occ.column\_value = mv.map\_key  AND occ.object\_package\_guid = mv.object\_package\_guid  WHERE occ.name = 'KEYWORD'  AND occ.object\_name = oc.object\_name  ) ca  WHERE (oc.object\_name = @TermDescription  OR oc.description LIKE '%' + ISNULL(@ColumnDesc,@TermDescription) + '%')  AND oc.column\_type = ISNULL(@ReadFlag, oc.column\_type)  ORDER BY EventName, column\_name;  GO  Basically, I am searching against the object\_columns metadata dmv for Extended Events just to find any columns that match my theoretical needs. In this case, I am trying to find anything that may help me find events related to the terms “select” or “statement”. When run, there will be quite a few results for these terms. That may actually be a bit of a problem (just the same as if nothing were returned in the results). There is a fix for that (well probably several fixes as you begin to tweak the query).  One possible fix is to also include map\_values to refine the search. Another fix might be to change the search terms. And still other fixes might include tweaking the predicate. For today, I am going to demonstrate the inclusion of map\_values as a secondary tweak to help isolate the needed event.  DECLARE @TermDescription VARCHAR(64) = 'Select'  , @ColumnDesc VARCHAR(64) = 'statement'  , @ReadFlag VARCHAR(64) = NULL; --readonly' --ALL if all columntypes are desired --data --customizable    SELECT oc.object\_name AS EventName  , oc.name AS column\_name  , oc.type\_name  , oc.column\_type AS column\_type  , oc.column\_value AS column\_value  , oc.description AS column\_description  , ca.map\_value AS SearchKeyword  , mv.map\_value AS MapValue  , mv.map\_key AS MapKey  , oc.column\_type  FROM sys.dm\_xe\_object\_columns oc  INNER JOIN sys.dm\_xe\_map\_values mv  ON oc.type\_name = mv.name  AND mv.object\_package\_guid = oc.object\_package\_guid  CROSS APPLY ( SELECT mv.map\_value  FROM sys.dm\_xe\_object\_columns occ  INNER JOIN sys.dm\_xe\_map\_values mv  ON occ.type\_name = mv.name  AND occ.column\_value = mv.map\_key  AND occ.object\_package\_guid = mv.object\_package\_guid  WHERE occ.name = 'KEYWORD'  AND occ.object\_name = oc.object\_name  ) ca  WHERE (oc.object\_name = @TermDescription  OR oc.description LIKE '%' + ISNULL(@ColumnDesc,@TermDescription) + '%')  AND mv.map\_value = @TermDescription  AND oc.column\_type = ISNULL(@ReadFlag, oc.column\_type);  In this iteration, my result-set is trimmed significantly due to the predicate and the requirement for a match based on the map\_value name. Ultimately, this trimmed my results down to precisely the events that I needed in this case. With this slightly more advanced version, I have greater visibility into the Extended Event realm to help me find just the right event to match my needs.  **Are there other Events related to the new found Event?**    This is some pretty awesome stuff so far. But it doesn’t end right there. Many events are tied together to help provide a more holistic view of the problem. These events are tied together via keyword. You can read more about keywords [here](http://bit.ly/1KDJrUP). By searching for events based on a keyword, we can get a lot more intelligent about the sessions we create with Extended Events.  Taking the single result from the last query and then using the SearchKeyword value, for that event, in this next query – I can potentially go from a Great DBA to a Super DBA.  DECLARE @Keyword VARCHAR(64) = 'execution' --'synchronization'    SELECT oc.OBJECT\_NAME AS EventName  ,oc.name AS column\_name, oc.type\_name  ,oc.column\_type AS column\_type  ,oc.column\_value AS column\_value  ,oc.description AS column\_description  ,REVERSE(LEFT(REVERSE(olm.name),CHARINDEX('\',REVERSE(olm.name))-1)) AS DLLName  ,olm.file\_version  ,xp.name AS PackageName  ,mv.map\_value AS SearchKeyword  ,ch.Channel  FROM sys.dm\_xe\_object\_columns oc  INNER JOIN sys.dm\_xe\_map\_values mv  ON oc.type\_name = mv.name  AND oc.column\_value = mv.map\_key  AND oc.name = 'KEYWORD'  INNER JOIN sys.dm\_xe\_packages xp  ON oc.object\_package\_guid = xp.guid  INNER JOIN sys.dm\_os\_loaded\_modules olm  ON xp.module\_address = olm.base\_address  INNER JOIN (SELECT c.object\_name AS EventName,c.object\_package\_guid AS PkgGuid, v.map\_value AS Channel  FROM sys.dm\_xe\_object\_columns c  INNER JOIN sys.dm\_xe\_map\_values v  ON c.type\_name = v.name  AND c.column\_value = CAST(v.map\_key AS NVARCHAR)  WHERE c.name = 'channel') ch  ON ch.EventName = oc.object\_name  AND ch.PkgGuid = oc.object\_package\_guid  WHERE mv.map\_value = @Keyword  ORDER BY EventName  ;  GO  I intentionally included the columns from each of the matching events. Doing this, I can also see what potential payload may be trapped when I attempt to trap any of the events from this result set.  **Conclusion**  There is great power in Extended Events. There is more power in being able to query the metadata for Extended Events in order to trap the appropriate information to properly troubleshoot a problem.  Have fun with these scripts and I hope you enjoy.  http://jasonbrimhall.info/?feed-stats-post-id=3559  **Related Posts:**   * [Extended Events – Events](http://jasonbrimhall.info/2015/09/10/extended-events-events/) September 10, 2015 * [Extended Event Help Queries](http://jasonbrimhall.info/2016/07/19/extended-event-help-queries/) July 19, 2016 * [Extended Events – Event Anatomy](http://jasonbrimhall.info/2015/09/11/extended-events-event-anatomy/) September 11, 2015 * [Shredding XML in XEvents](http://jasonbrimhall.info/2015/08/26/shredding-xml-in-xevents/) August 26, 2015 * [Map Data Types in Extended Events](http://jasonbrimhall.info/2015/09/18/map-data-types-in-extended-events/) September 18, 2015 |

|  |
| --- |
|  |