IF EXISTS(SELECT \* FROM sys.server\_event\_sessions WHERE name='LongRunningQuery')

DROP EVENT SESSION LongRunningQuery ON SERVER

GO

-- Create Event

CREATE EVENT SESSION LongRunningQuery

ON SERVER

-- Add event to capture event

ADD EVENT sqlserver.rpc\_completed

(

ACTION (sqlserver.sql\_text, sqlserver.tsql\_stack, sqlserver.client\_app\_name,

sqlserver.username, sqlserver.client\_hostname, sqlserver.session\_nt\_username)

WHERE (

duration > 1000

AND sqlserver.client\_hostname <> 'A'

)

),

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION (sqlserver.sql\_text, sqlserver.tsql\_stack, sqlserver.client\_app\_name,

sqlserver.username, sqlserver.client\_hostname, sqlserver.session\_nt\_username)

WHERE (

duration > 1000

AND sqlserver.client\_hostname <> 'A'

)

),

ADD EVENT sqlserver.module\_end

(

ACTION (sqlserver.sql\_text, sqlserver.tsql\_stack, sqlserver.client\_app\_name,

sqlserver.username, sqlserver.client\_hostname, sqlserver.session\_nt\_username)

WHERE (

duration > 1000000

AND sqlserver.client\_hostname <> 'A'

)

)

ADD TARGET package0.asynchronous\_file\_target(

SET filename='C:\New folder\LongRunningQuery.xet', metadatafile='C:\New folder\LongRunningQuery.xem'),

ADD TARGET package0.ring\_buffer

(SET max\_memory = 4096)

WITH (max\_dispatch\_latency = 1 SECONDS, TRACK\_CAUSALITY = ON)

GO

-- Enable Event,

ALTER EVENT SESSION LongRunningQuery ON SERVER

STATE=START

GO

DECLARE @XMLLongRunning XML

SELECT @XMLLongRunning = CAST(dt.target\_data AS XML)

FROM sys.dm\_xe\_session\_targets dt

JOIN sys.dm\_xe\_sessions ds

ON ds.Address = dt.event\_session\_address

JOIN sys.server\_event\_sessions ss

ON ds.Name = ss.Name

WHERE dt.target\_name = 'ring\_buffer'

AND ds.Name = 'LongRunningQuery'

select T.N.value('local-name(.)', 'varchar(max)') as Name,

T.N.value('.', 'varchar(max)') as Value

from @XMLLongRunning.nodes('/\*/@\*') as T(N)

-- Stop the event

ALTER EVENT SESSION LongRunningQuery ON SERVER

STATE=STOP

GO

-- Clean up. Drop the event

DROP EVENT SESSION LongRunningQuery

ON SERVER

GO

------------------------------

--Shred XML for easy reading--

------------------------------

--pull into temp table for speed and to make sure the ID works right

if object\_id('tempdb..#myxml') is not null

DROP TABLE #myxml

CREATE TABLE #myxml (id INT IDENTITY, actual\_xml XML)

INSERT INTO #myxml

SELECT CAST(event\_data AS XML)

FROM sys.fn\_xe\_file\_target\_read\_file

('C:\New folder\LongRunningQuery\*.xet',

'C:\New folder\LongRunningQuery\*.xem',

NULL, NULL)

--Now toss into temp table, generically shredded

if object\_id('tempdb..#ParsedData') is not null

DROP TABLE #ParsedData

CREATE TABLE #ParsedData (id INT, Actual\_Time DATETIME, EventType sysname, ParsedName sysname, NodeValue VARCHAR(MAX))

INSERT INTO #ParsedData

SELECT id,

DATEADD(MINUTE, DATEPART(TZoffset, SYSDATETIMEOFFSET()), UTC\_Time) AS Actual\_Time,

EventType,

ParsedName,

NodeValue

FROM (

SELECT id,

A.B.value('@name[1]', 'varchar(128)') AS EventType,

A.B.value('./@timestamp[1]', 'datetime') AS UTC\_Time,

X.N.value('local-name(.)', 'varchar(128)') AS NodeName,

X.N.value('../@name[1]', 'varchar(128)') AS ParsedName,

X.N.value('./text()[1]', 'varchar(max)') AS NodeValue

FROM [#myxml]

CROSS APPLY actual\_xml.nodes('/\*') AS A (B)

CROSS APPLY actual\_xml.nodes('//\*') AS X (N)

) T

WHERE NodeName = 'value'

DECLARE @SQL AS VARCHAR (MAX)

DECLARE @Columns AS VARCHAR (MAX)

SELECT @Columns=

COALESCE(@Columns + ',','') + QUOTENAME(ParsedName)

FROM

(

SELECT DISTINCT ParsedName

FROM #ParsedData

WHERE ParsedName <> 'tsql\_stack'

) AS B

SET @SQL='

SELECT Actual\_Time, EventType,' + @Columns + ' FROM

(

SELECT id, EventType, Actual\_Time, ParsedName, NodeValue FROM

#ParsedData ) AS source

PIVOT

(max(NodeValue) FOR source.ParsedName IN (' + @columns + ')

)AS pvt order by actual\_time, attach\_activity\_id'

EXEC (@sql)