Extended Events Step By Step

# Introduction

I’ve read a few articles about using Extended Events (aka xEvents) to perform troubleshooting, and they do indeed provide a very powerful tool to troubleshoot. This is particularly true in Azure DB where in many cases we don’t have many alternatives, there is no Profiler (or as we’d prefer server side traces) so we don’t get much in the way of choice. There are a number of great articles on how to troubleshoot a specific issue with extended events, but so far I’ve not found an end to end, step by step guide to take a user through from start to finish, so as with many things in life, if you want a job done right, you have to do it yourself. I am going to try and do this using only the management portal, and SQL Server Management Studio, if you are thinking ‘wow we could do that in Power Shell’ chances are you don’t need the help below.

# Capturing Extended Events

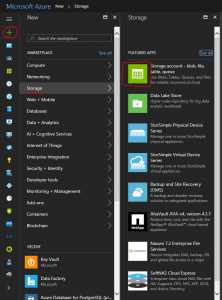
When working with extended events there are two places you can capture the events, the first is the ‘ring buffers’ this is in-memory, and obviously the amount of space we can use for this is going to be limited, the other drawback is that its not very portable or easy to share. The alternative is a file which is great for on-prem, but in Azure we have to place this file in a storage account which adds a level of complexity for us.  
The overall steps for creating and capturing extended events from an Azure DB are going to be as follows

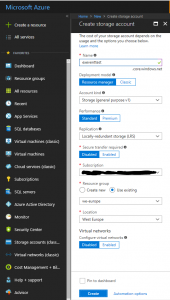
* Set up a storage account and container
* On the Azure DB server create a master encryption key
* Create credentials in Azure DB to access the storage account
* Configure Extended event session
* Run the capture

I am going to assume that there is nothing set up for us and we are starting from scratch.

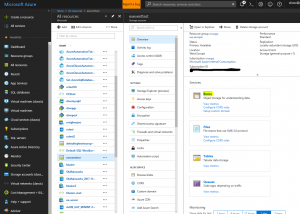
# Setting up a storage account

OK so on the Portal we are going to select New Storage  
http://portal.azure.com/

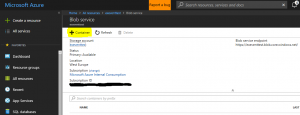
[](https://msdnshared.blob.core.windows.net/media/2018/03/fig1.png)

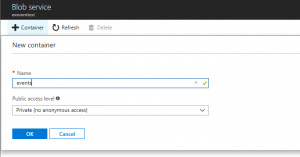
[](https://msdnshared.blob.core.windows.net/media/2018/03/fig2.png)

This should then spin away deploying for us, it will take a little while but after we should have the storage account pinned to our dashboard.  
Once we have an account created we need to make a container to put our file in. On the dashboard select the storage account and the under the Blob Storage, select Containers

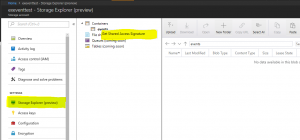
[](https://msdnshared.blob.core.windows.net/media/2018/03/fig3.png)

You can then press + Container and add your container. I’ve created one called events here.

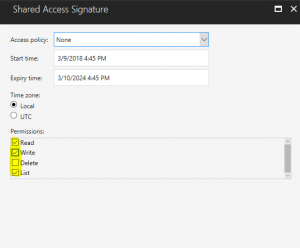
[](https://msdnshared.blob.core.windows.net/media/2018/03/Fig4.png)

[](https://msdnshared.blob.core.windows.net/media/2018/03/Fig5.png)

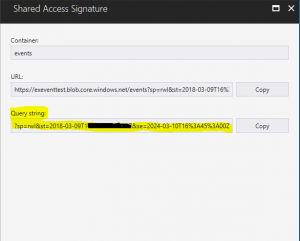
To ensure that access to our storage is secure we must create a SAS (Shared Access Signature) key. I did this in the Storage Explorer in the Azure Portal.

[](https://msdnshared.blob.core.windows.net/media/2018/03/Fig6.png)

***The Permissions are very important***!

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig7.png)

This should then provide is wit the key for the events container, which we can copy and use later.

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig8.png)

# Setting up the Extended Events

OK so that is the first phase of the operation, we now have a storage account and container ready for our extended event capture.

## Create the Credentials

Meanwhile in SQL Server Management Studio we first need to create some credentials to access the storage account.  
Step 1 is to create a secured master key (if you don’t have one already. Use the following command to do this.

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'x4565465xxxxx!';

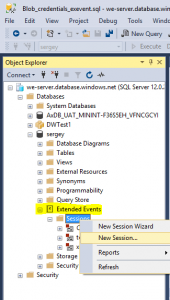
Then go ahead and create the Database Scoped Credential in the database you are going to monitor

CREATE DATABASE SCOPED CREDENTIAL [https://exeventtest.blob.core.windows.net/events]  
WITH IDENTITY='SHARED ACCESS SIGNATURE',  
SECRET = 'sp=rwl&st=2018-03-09T16%3A45%3A00Z&se=2024-03-10T1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxWM0%3D&sr=c'

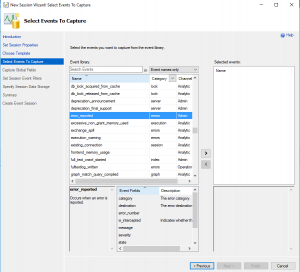
## NOTE: You must remove the leading ‘?’ from the key

Creating the session

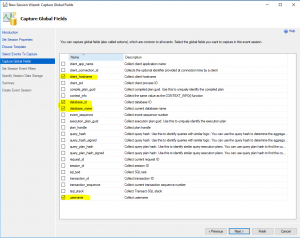
Now we can create a session, to do this via SSMS we expand the database and open Extended Events then right click on Sessions, there is then to option to create a new session via a wizard or manually.

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig9.png)

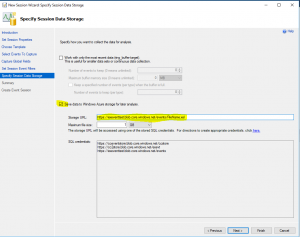
In the Wizard we first select the Session name and if we want to use a template, I’ve skipped those steps and moved to the events to capture.  
For this example, I am going to show how to capture errors, so the event I select is the error\_reported event

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig10.png)

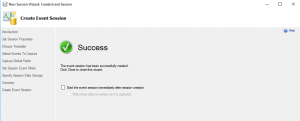
Then some additional data I may need, here I select the database\_id , client\_hostname, and username

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig11.png)

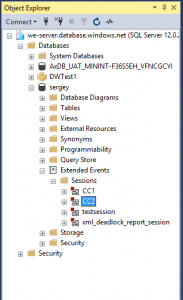
I am not filtering at this point so skip ahead to storage

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig12.png)

And we are done we can hit Finish…. And you should end up with the following.

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig14.png)

Hit Close and refresh the Object Explorer.

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig15.png)

And there it is...

# Capturing a session.

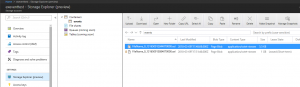
The first thing to do is to start the session. You should see the red square turn into a play button

[https://msdnshared.blob.core.windows.net/media/2018/03/missing2.png](https://msdnshared.blob.core.windows.net/media/2018/03/missing2.png)

Now to make an error, run the following in your database

select wibble from wobble

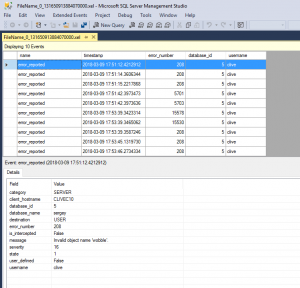
In the storage explorer for the blob container you should now see a file(s) (I had a couple of goes at this)

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig16.png)

Once you stop the session the lock will be removed from the file and you can then open the file and marvel at its contents.

# Inspecting the Capture

I downloaded the file and then opened it in SSMS

[](https://msdnshared.blob.core.windows.net/media/2018/03/fig17.png)

The example given here is not the most complex but it does offer an end to end walkthrough, there are many different scenarios where Extended events can be used in SQL and Azure DB.  
As well as using SSMS to inspect the contents, it is also possible to dive into them directly.  
The internet has many examples of capturing extended events , but hopefully this will give an end to end example which can aid you when working through those examples.