BRENT OZAR
UNLIMITED®

The Consultant Toolkit

Saturday, September 4, 2021

Getting support

Email us at Help@BrentOzar.com whenever you run into issues. Our email address isn't in the zip file because I'm sure some consulting shops will just take the zip file exactly as-is, and email it to customers. We want your customers to stay your customers, and we don't wanna know who they are.

How to run the app

Requirements

- SQL Server 2008 or newer
- .NET 4.5.2 or higher
- Sysadmin permissions – because to query for things like CHECKDB success, you have to be SA. The app will run with lower permissions like VIEW SERVER STATE, and it'll throw warnings when it's unable to query those SA-only functions, but you'll still get a ton of usable diagnostic data.
- Cloud compatibility:
    - Amazon RDS SQL Server – fully supported
    - Azure Managed Instances – fully supported
    - Azure SQL DB and Hyperscale – supported as long as you pass in a database name in the command line as shown in the readme.txt. sp_Blitz will not run, and will throw a yellow error – that's because Azure SQL DB doesn't support master system object queries yet.

Download it to your desktop and extract it to your local drive, like c:\temp. You don't have to run this on the SQL Server itself - you can run it from any desktop or laptop with at least .NET 4.5.2. Then, to gather diagnostic data on one of your servers, go to a command prompt and run:

Windows authentication:

```
AppName.exe --datasource MyServerName
```

SQL Server authentication:

```
AppName.exe --datasource MyServerName --userid MyUserName --password
MyPassword
```

If you use tricky characters in your server name, user name, or password, like quotation marks or spaces, you'll need to surround it with double quotes. For example, if my password is @#!, I might want to use "@#!".

It'll take about 5-15 minutes to run a bunch of DMV queries, and when it's done, check out the Output folder. There's a zip file with your server's data.

To see a video of me walking through the process:

https://www.youtube.com/watch?v=2MdBiu1BQxs

By default, it runs a relatively limited set of queries. When you're working hands-on with a client's server yourself, and you want even more, add --deepdive (that's two dashes, not one) as a parameter, and it'll run more:

```
AppName.exe --datasource MyServerName --deepdive
```

Running it in Deep Dive will take longer, though, since it runs more queries.

If you go into the Resources folder, you'll see the queries it runs: things like sp_Blitz, sp_BlitzCache, sp_BlitzIndex, and other utility queries we've written over the years.

Psst: the real name isn't AppName.exe.

The real app name is shown inside the zip file, but I'm purposely not mentioning it here, dear reader. See, if you're a consultant, you're going to be giving this app to your clients to run for diagnostic purposes. I don't want your clients Googling the app's name and finding me. This is *your tool,* the thing that's gonna give you an edge over the amateurs who are still copy/pasting data out of SSMS.

There is a support web site with more details inside the readme.txt – that site is designed for you and your customers, including detail about the queries it runs.

Errors? Don't panic.

This app runs database queries, and...sometimes queries return unexpected results. After all, the client's in trouble - that's why they're calling you! If the app gets any errors, they'll be logged to the .log text file inside the output.

If you've got error questions, email us the log file at Help@BrentOzar.com. (You don't have to send us the whole zip file - just the log is fine.)

A couple of known issues:

- If you get a warning when opening the Excel spreadsheet, that's fine and ignorable. The app takes data from your plan cache and puts it straight into Excel cells, so from time to time, you'll have query contents that will give Excel problems when it opens the file. Just say yes, open the file anyway, and Excel will prompt you to delete the discarded contents. That's okay – the spreadsheet is still chock full of diagnostic goodness.
- Compatibility with Azure SQL DB, Managed Instances, and Hyperscale can change at a moment's notice. Microsoft has been known to change availability of system views without warning.

Running it under a different domain account

Say you're connected to a VPN with Windows authentication, and you need to run the app to connect to SQL Server using someone else's domain account:

```
C:\Windows\System32\runas.exe /user:{domain}{username} /netonly
"{install directory}\APPNAME.exe --datasource {servername}"
```

For more information about how to use runas, see the runas documentation.

Customizing the spreadsheet with your logo

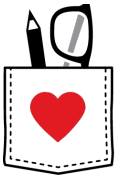
You can do anything you want on the first tab of the spreadsheet (Introduction), including changing the tab name, deleting the contents, organizing them completely differently, absolutely anything you want.

Whenever you download a new version of the app, you can simply copy your customized sheet into the new version's spreadsheet.

However, do not make changes to anything but the first tab. The other tabs are very specific: the columns and rows can't change. (If you read querymanifest.json, you'll see how the data is mapped from query results into the spreadsheet using rows & columns.)

Sharing the app with your team & clients

This app has a built-in expiration: it expires 90 days after it's compiled. That's our way of ensuring 3 things:



1. You've always got the latest diagnostic scripts
2. You've got a current subscription
3. Your clients aren't reusing this for years after you leave – see, I've learned that this app is a huge value-add for clients, and your clients are going to fall in love with how easy it is to get this diagnostic data. I want them to keep coming back to you for help, and not just take this app and wave goodbye to you.

That means when you're designing your company processes around how you'll share this app with your team & your clients, keep in mind that it'll need to be replaced on a regular basis.

Put it in an easily accessible place, like on your company's web site (but not publicly linked, since you don't want everybody getting a hold of this.)

Uploading the results to Amazon S3

You can configure the .exe.config file with AWS credentials to have the results automatically uploaded to an S3 bucket whenever --upload is used on the command line. This is helpful when you want to set up a scheduled task at the client's site to automatically send you diagnostic data every week (or day) to keep tabs on how their server is doing.

Here's how to set it up:

https://www.brentozar.com/training/consultant-toolkit/uploading-to-amazon-s3/

Email template for sales calls – adapt with your URL

Hi! You have just one step left before our sales call. Download our free utility and run it on your desktop:

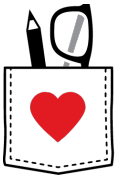
http://notarealwebsite.com/DataCollector.zip

Extract that in an easy-to-find folder like C:\temp, and then open a command prompt, navigate to that folder, and type:

AppName.exe -s MYSERVERNAME

If you use SQL authentication to connect to the server, you can specify a username & password:

AppName.exe -s MYSERVERNAME -i MYUSERNAME -p MYPASSWORD



BRENT OZAR
UNLIMITED

It takes about 5-15 minutes to run a bunch of low-overhead, low-priority queries and packages the results up into a zip file. It doesn't change anything on your server or block queries – it just gathers metrics. I can slice and dice those ahead of our call to give you even better diagnostics. If you get any errors in the app, absolutely no worries – they'll be logged and sent to me too.

After it runs, go into the Output folder and email me the zip file. (If you're curious about what's in it, you're welcome to look at the results – the more you know about your SQL Server, the better!)

Talk to you soon!

Customizing the timeouts and the queries it runs

We don't officially support this yet, because it'll be kinda painful for you to manage each time we bring out a new version. (Remember, the app has a 90-day time limit built in because we want to make sure everybody's on the latest version of the diagnostic queries.)

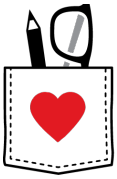
You could technically do it by going into the Resources folder and editing the querymanifest.json file, and editing the query files and timeouts. If you do that, just be mindful to only add NEW queries, or REMOVE existing queries completely. That'll make it much easier for you to deal with the differences every 90 days.

Now that we're selling this app to the public, this is probably the first feature on the roadmap, though. We haven't done design on this yet, but my guess is that we're going to let you add a querymanifest_custom.json or something to that effect – where you can add your own queries. That way, whenever you download a new version from us, you'd just copy in your querymanifest_custom.json, and off you go.

How I review results

Alright, let's set expectations here: if I could make you an expert at SQL Server in a single web page, I wouldn't sell a Live Class Season Pass. But in one web page, let's see if I can distill the process I use to analyze a client's diagnostic metrics.

I work through the first few tabs of the spreadsheet in order:



First, I look at the Uptime tab. In just a few metrics, I can get a pretty good idea of whether I'm dealing with an overworked monster, or a bored desktop, or a server who has amnesia because she just got rebooted and lost all her metrics.

Next, I hit the Health tab, sp_Blitz's results. I want to know if the databases are backed up because I'm a little paranoid: I won't work on a server without backups. Scrolling down through the rest of the results gives me a quick idea of whether the server has basic best practices set up (like Cost Threshold and MAXDOP configured, or whether auto-shrink has been turned on.)

Then, I check the Waits tab. I'm looking at wait time ratio, the amount of uptime versus time we've spent waiting on stuff. A couple of simple examples:

- 100 hours since startup, but only the top wait is only 10 hours – the server is bored.
- 100 hours since startup, but the top wait is over 10,000 hours – the server is busting its hump.

If the server is bored, that doesn't mean every query is fast – but it means that I'm probably not going to be solving a server-level problem, like not enough CPUs or memory.

In short:

- The Uptime tab tells me about the hardware
- The Health tab tells me about the people managing it
- The Waits tab tells me whether the users are happy

Then, based on what their top waits are, I branch off to looking at indexes, queries, storage performance, etc.

Getting support

Email us at Help@BrentOzar.com whenever you run into issues. Thanks, and hope this makes you a more powerful, productive consultant!

Brent