BACPAC using SqlPackage

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:25 AM PST

Contents

How To: BACPAC using SqlPackage

When you need to export a database for archiving or for moving to another platform, you can export the database schema and data to a BACPAC file. A BACPAC file is a ZIP file with an extension of BACPAC containing the metadata and data from a SQL Server database. A BACPAC file can be stored in Azure Blob storage or in local storage in an on-premises location and later imported back into Azure SQL Database or into a SQL Server on-premises installation.

SqlPackage is the recommended utility when the Import/Export in the Azure Portal cannot be used, or when we need more data for troubleshooting.

All examples will use Powershell, but can easily be adapted to either Az CLI or even Bash for customers using Linux Virtual Machines. I recommend installing the .Net Core versions of SqlPackage, since they are contained in a single folder and can be easier to manage for customers who are unable to install new software in Production machines.

Download SqlPackage

The first section will contain a full script to Export a database from Azure SQL PaaS, using SQL authentication.

```
# Server/Database
$server_name = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.database.windows.net'
$database_name = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

# Credentials
$user_name = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
$password = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

# File location
$date = Get-Date -Format "yyyyMMdd_HHmmss"
$storage = "C:\BACPAC"
$sqlpackage = $($Storage + '\sqlpackage\sqlpackage.exe')
$source_file = $($Storage + '\' + $database_name)
$target_file = $($Storage + '\' + $database_name + '_' + $date)
$log_file = $($Storage + '\' + $database_name + '_' + $date)


# Actions
& $sqlpackage /a:Export /ssn:$server_name /sdn:$database_name /tf:$target_file'.bacpac' /df:$log_file'.log' /s
```

Notes

- When possible use the parameter **DiagnosticsFile** *(df)* to specify a file where diagnostics are saved to, as part of best practices.
- Date is being appended to file names, allowing the same script to be executed multiple times without losing previous results.
- Ensure there is enough storage space in OS drive, because temporary files are saved there by default. This can be changed on Windows settings if needed.
- The action show above was Export, for other actions please ensure you are using the corresponding parameters *(e.g.: SourceServerName instead of TargetServeName)*. See below an example for Import action.
- BACPAC exports *are not transactionally consistent*, write activity during an Import/Export can lead to errors

```
& $sqlpackage /a:Import /tsn:$server_name /tdn:$database_name /sf:$source_file'.bacpac' /df:$log_file'.log' /a
```

Token based AD Authentication

When customer want to use token based authentication instead of the standard SQL auth, the access token must be retrieved first from its appropriate endpoint and then used in the SqlPackage parameters **AccessToken** *(at)*.

This type of authentication uses Azure Active Directory, see Azure Active Directory authentication

- **Service Principal**

With this method, we must provide all the necessary information about the Active Directory Tenant and the principal's unique identifier and secret/password. With it, we can then request the access token.

```
# AD Details
$tenant = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
$SPN = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
$secret = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

# Convert the Secret to a secure string a create a Credential objet
$secure_secret = ConvertTo-SecureString $secret -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential ($SPN, $secure_secret)

# Connect the SPN and grab an access token
$Account = Connect-AzAccount -ServicePrincipal -Credential $credential -Tenant $tenant
$AccessToken = (Get-AzAccessToken -ResourceUrl https://database.windows.net).Token
```

- **Managed Identity**

This method is easier to setup, because all the information is coming from the resource *(e.g.: Azure VM)* that is requesting the access token.

```
$HEADERS=@{Metadata="true"}
$URI = "http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://database
$AccessToken = $(Invoke-RestMethod -Method 'Get' -Uri $URI -Headers $HEADERS).access_token
```

External Links

SqlPackage

Use Azure Active Directory authentication

Tutorial: Use a Windows VM system-assigned managed identity to access Azure SQL

How good have you found this content?