Microsoft

SQL Server to Azure Synapse Migration Process, Modules and Scripts

Prepared by

Gaiye “Gail” Zhou

Contributors

Faisal Malik, Aruna Dadi

Azure Data & AI Americas

Version: Draft V1.0

May 2021

Contents

Disclaimer

1 Design Principles and Programming Styles

1.1 Overview

What does the utilities do?

- ✓ Translate SQL Server Table DDLs into Azure Synapse DDLs
- ✓ Execute Translated Table DDLs in Azure Synapse (Code Migration)
- ✓ Export SQL Server Data
- ✓ Upload Exported Data into Azure Data Lake Store (or Blob Storge)
- ✓ Generate T-SQL Copy Import Scripts
- ✓ Execute T-SQL Copy Import Scripts to Import Data into Azure Synapse

Why do we need this utility when we already have Azure Synapse Pathway?

This utility is complementary to Azure Synapse Pathway (ASP). ASP does not perform data migration today. We designed and implemented 5 modules to complete the end-to-end tasks of tables migration and data migration. Please check the latest release of Azure Synapse Pathway for more advanced SQL Server code translation capabilities so you can use the best available functions: sql-docs/azure-synapse-pathway-overview.md at live · MicrosoftDocs/sql-docs (github.com)

In addition, you can use scripts described in this document to execute all translated code by Azure Synapse Pathway or other methods. Module 3, 4, 5 are reusable for other types of migrations, for example, Netezza or Teradata to Azure Synapse migrations. After the code is translated, and data is exported out of source systems, the rest of the tasks are the same. Therefore module 3-5 can be utilized for any of those migrations.

1.2 Design Principles

We adhere to below design principles:

- ✓ **Modular**: Modules that run *independently* but can use output from other modules
- ✓ **Consistent**: All driver programs are written in PowerShell Scripts.
- ✓ **Simple**: Only one PowerShell Program (scripts) for each module.
- ✓ **Configurable**: Each module has easy way to config parameters.
- ✓ **User Friendly**: Users are prompted for config file name only. Well documented configuration parameters. Strong error handling mechanism to provide friendly messages. Sample config files are provided. Manual work is minimalized. Utilities are provided to generate config file(s) that involves list of tables.
- ✓ **Reusable**: Module 1 is for code translation; Module 2 is for SQL Server data export. Only these two modules are specific to SQL Server. *Modules 3-5 are reusable for any migration into Azure Synapse, the sources can be: Netezza, Teradata, Exadata, Oracle, DB2, Snowflake, Redshift, Google Big Query, etc., once the code is translated and data is exported out of source system.*
- ✓ **Extensible to Leverage Azure Synapse Pathway:** Current version only translates tables. When Azure Synapse Pathway releases the full version of code translation from SQL server to Azure Synapse, the translated code can be readily utilized by this process. You just need execute translated code using Module 5, "5_*RunSqlFilesInFolder*". You only need to specify the folder where the Translated Code is stored.

1.3 Best Practices in Programming Styles

We adopt below best practice as our programming style:

- ✓ Being Protective - No hardcoded security information anywhere. We will ask you to provide security such as username and password.
- ✓ Being Assertive – We will ask you to specify location of needed software such as BCP or Azcopy.
- ✓ Being Friendly - We prompt you for your information with sample values. We also provide utilities to generate configuration files.

2 Overview – Modules and Scripts

There are five modules that contain PowerShell Scripts and T-SQL Scripts designed to accomplish key task(s) that are relevant to SQL server to Azure Synapse migration.

The five modules are summarized as below:

1_TranslateMetaData: Translate SQL objects (DDLs) from source system format to Azure Synapse format. The output is stored as .sql files in specified file folder (configurable).

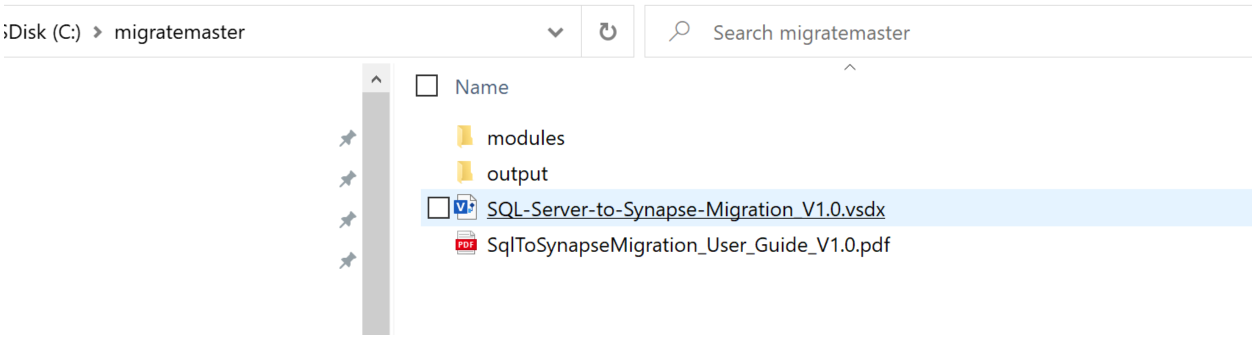
2_ExportSourceData: Export SQL Server Tables into data files stored in predefined structure and format (.csv).

3_LoadDataIntoAzureStorage: Load exported data files into specified container in Azure Storage (Blob Storage or Azure Data Lake Store).

4_GenerateCopyIntoScripts: Generate "COPY Into" T-SQL Scripts that will move data from Azure Storage into Azure Synapse SQL Pool tables, once executed.

5_RunSqlFilesInFolder: Run all T-SQL Scripts defined in .sql files stored in a specified file folder. The T-SQL Scripts can be DDL, DML, Data Movement Scripts (such as Copy Into scripts), or any other scripts such as create/update statistics or indexes. In fact, this module is designed to run any SQL scripts in a folder.

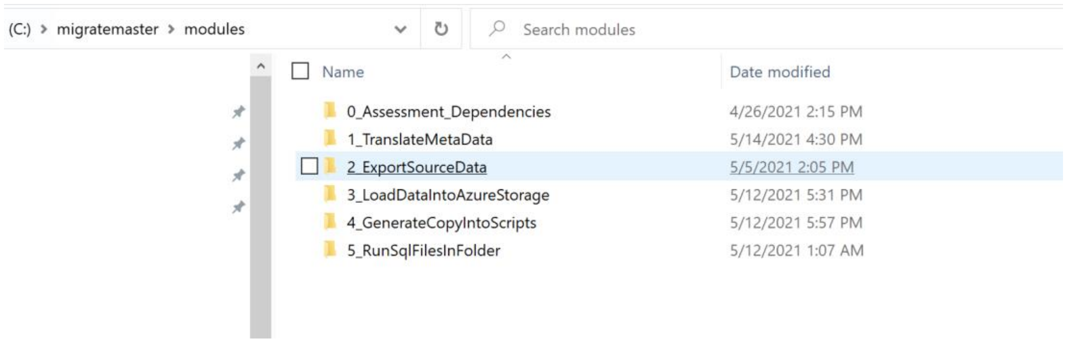
The organization of the modules, output folder, and documentations is illustrated in Figure 1. Modules are stored in the "modules" directory. "output" is designed to store output of the modules. You can use different folder as you wish. It is configurable.



modules
output
SQL-Server-to-Synapse-Migration_V1.0.vsdx
SqlToSynapseMigration_User_Guide_V1.0.pdf

Figure 1 File Structures and Documentations

Inside the "modules" directory, there are 5 modules, as illustrated in Figure 2.



Name	Date modified
0_Assessment_Dependencies	4/26/2021 2:15 PM
1_TranslateMetaData	5/14/2021 4:30 PM
2_ExportSourceData	5/5/2021 2:05 PM
3_LoadDataIntoAzureStorage	5/12/2021 5:31 PM
4_GenerateCopyIntoScripts	5/12/2021 5:57 PM
5_RunSqlFilesInFolder	5/12/2021 1:07 AM

Figure 2 Organization of Modules, output folder and documentations

3 How Migration Tasks are Modularized

Migration process is illustrated in Figure 3. The numbers specify the module numbers, not the execution sequence.

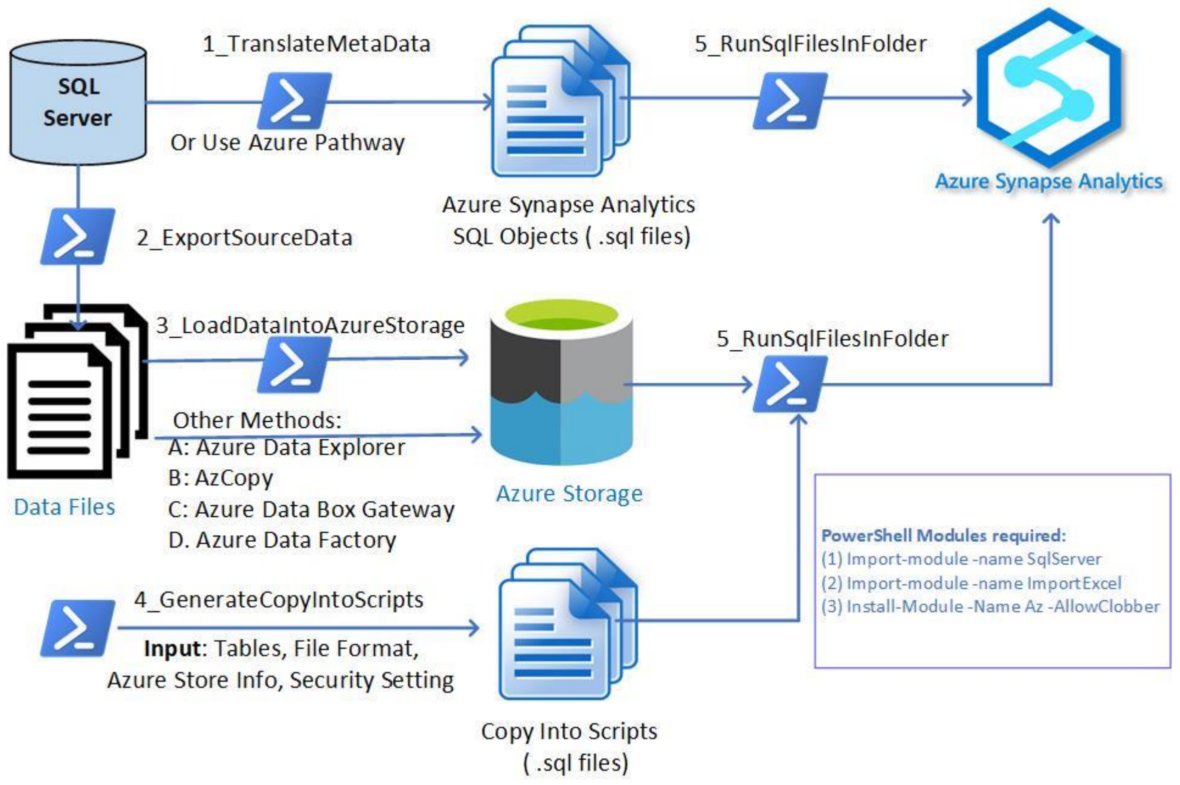

SQL Server
1_TranslateMetaData
Or Use Azure Pathway
Azure Synapse Analytics SQL Objects (.sql files)
5_RunSqlFilesInFolder
Azure Synapse Analytics
2_ExportSourceData
3_LoadDataIntoAzureStorage
Other Methods:
A: Azure Data Explorer
B: AzCopy
C: Azure Data Box Gateway
D. Azure Data Factory
Data Files
Azure Storage
5_RunSqlFilesInFolder
PowerShell Modules required:
(1) Import-module -name SqlServer
(2) Import-module -name ImportExcel
(3) Install-Module -Name Az -AllowClobber
4_GenerateCopyIntoScripts
Input: Tables, File Format, Azure Store Info, Security Setting
Copy Into Scripts (.sql files)

Figure 3 How Migration IP Modules are Applied for the Migration Process

Source System Tasks (SQL Server):

- Use **1_TranslateMetadata** to translate and save results into .sql files
- Use **2_ExportSourceData** to export source data and save results into local storage.

Data Movement Tasks:

- Use **3_LoadDataIntoAzureStorage** to load exported data into Azure Storage. Optionally you can use AzCopy or Azure Data Box Gateway to complete the task. Azure Data Box Gateway is a good choice if the volume of data is very large, for example, 50TB+.
- Use **4_GenerateCopyIntoScripts** to prepare "Copy Into" T-SQL Scripts
- Use **5_RunSqlFilesInFolder** to execute "Copy Into" T-SQL Scripts

Target System Tasks (Azure Synapse):

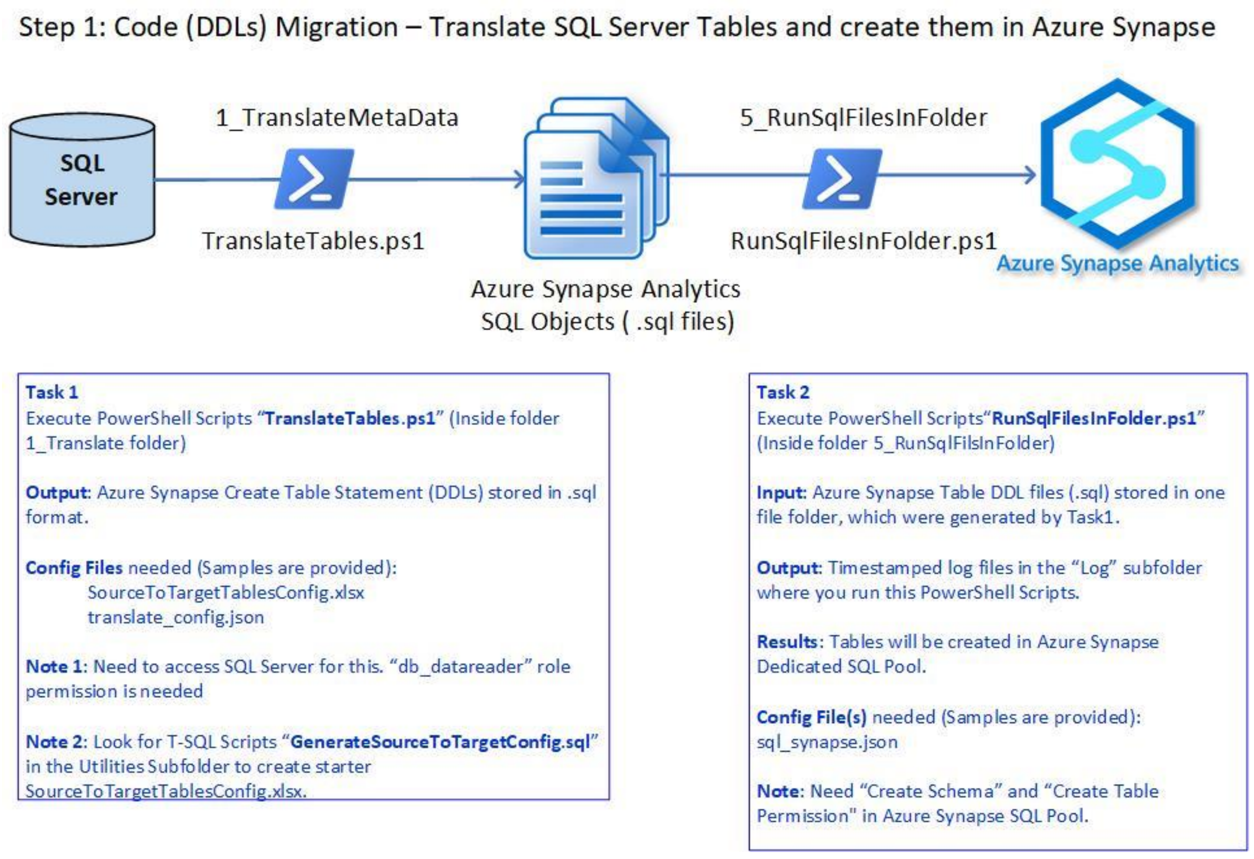
- Use **5_RunSqlFilesInFolder** to execute Azure Synapse T-SQL Scripts prepared by **1_TranslateMetadata,** to create meta data (tables) in Azure Synapse
- Use **5_RunSqlFilesInFolder** to execute "Copy Into" T-SQL Scripts prepared by 4**_GenerateCopyIntoScripts,** to move data from Azure Storage into Synapse

4 Step-by-Step Migration Guide

Although some of the module can be run in parallel, we have design a simple sequence for your Migration Journey.

4.1 Step 1 - Code (DDLs) Migration

Tables migration process is illustrated in Figure 4. For details on how to run each module, please refer to Section 5, Detailed Reference Guide for Each Module.


Step 1: Code (DDLs) Migration – Translate SQL Server Tables and create them in Azure Synapse
SQL Server
1_TranslateMetaData
TranslateTables.ps1
Azure Synapse Analytics SQL Objects (.sql files)
5_RunSqlFilesInFolder
RunSqlFilesInFolder.ps1
Azure Synapse Analytics

Task 1
Execute PowerShell Scripts "**TranslateTables.ps1**" (Inside folder 1_Translate folder)

Output: Azure Synapse Create Table Statement (DDLs) stored in .sql format.

Config Files needed (Samples are provided):
SourceToTargetTablesConfig.xlsx
translate_config.json

Note 1: Need to access SQL Server for this. "db_datareader" role permission is needed

Note 2: Look for T-SQL Scripts "**GenerateSourceToTargetConfig.sql**" in the Utilities Subfolder to create starter SourceToTargetTablesConfig.xlsx.

Task 2
Execute PowerShell Scripts"**RunSqlFilesInFolder.ps1**" (Inside folder 5_RunSqlFilsInFolder)

Input: Azure Synapse Table DDL files (.sql) stored in one file folder, which were generated by Task1.

Output: Timestamped log files in the "Log" subfolder where you run this PowerShell Scripts.

Results: Tables will be created in Azure Synapse Dedicated SQL Pool.

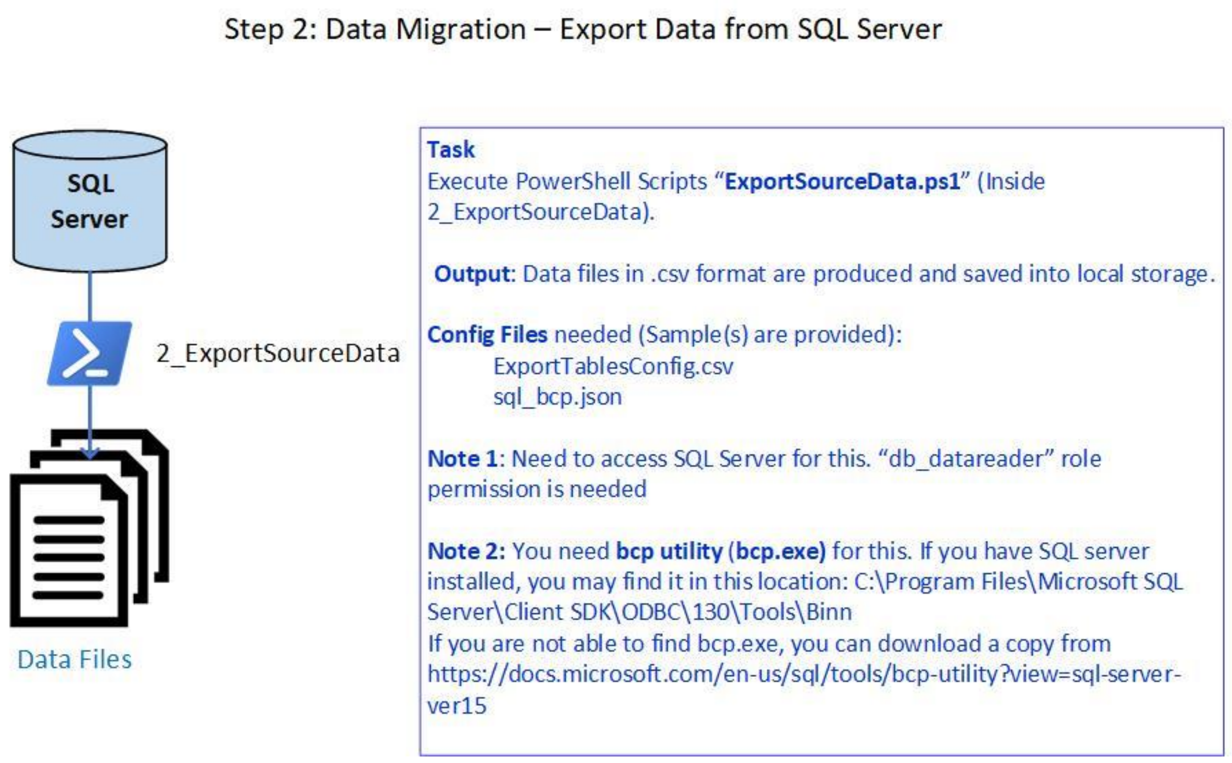
Config File(s) needed (Samples are provided):
sql_synapse.json

Note: Need "Create Schema" and "Create Table Permission" in Azure Synapse SQL Pool.

Figure 4 Step 1: Code (DDLs) Migration

4.2 Step 2 - Export Data out of SQL Server

Exporting SQL Server data process is illustrated in Figure 5. For details on how to run each module, please refer to Section 5, Detailed Reference Guide for Each Module.



Task
Execute PowerShell Scripts "**ExportSourceData.ps1**" (Inside 2_ExportSourceData).

Output: Data files in .csv format are produced and saved into local storage.

Config Files needed (Sample(s) are provided):
ExportTablesConfig.csv
sql_bcp.json

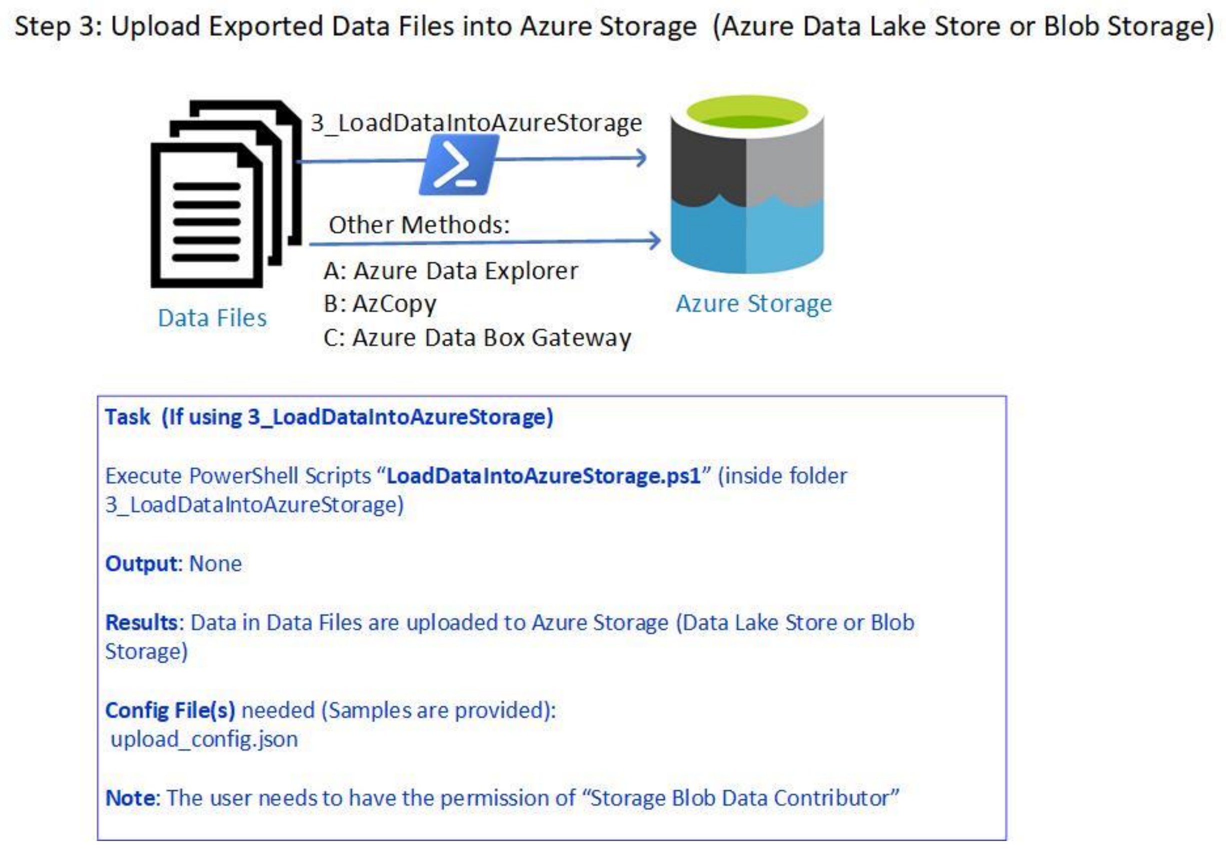
Note 1: Need to access SQL Server for this. "db_datareader" role permission is needed

Note 2: You need **bcp utility (bcp.exe)** for this. If you have SQL server installed, you may find it in this location: C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn
If you are not able to find bcp.exe, you can download a copy from https://docs.microsoft.com/en-us/sql/tools/bcp-utility?view=sql-server-ver15

Figure 5 Step 2: Export SQL Server Data

4.3 Step 3 – Upload Data into Azure Data Lake Store or Blob Storage

Uploading data into Azure Storage process is illustrated in Figure 6. For details on how to run each module, please refer to Section 5, Detailed Reference Guide for Each Module


Step 3: Upload Exported Data Files into Azure Storage (Azure Data Lake Store or Blob Storage)

Task (If using 3_LoadDataIntoAzureStorage)

Execute PowerShell Scripts "**LoadDataIntoAzureStorage.ps1**" (inside folder 3_LoadDataIntoAzureStorage)

Output: None

Results: Data in Data Files are uploaded to Azure Storage (Data Lake Store or Blob Storage)

Config File(s) needed (Samples are provided):
upload_config.json

Note: The user needs to have the permission of "Storage Blob Data Contributor"

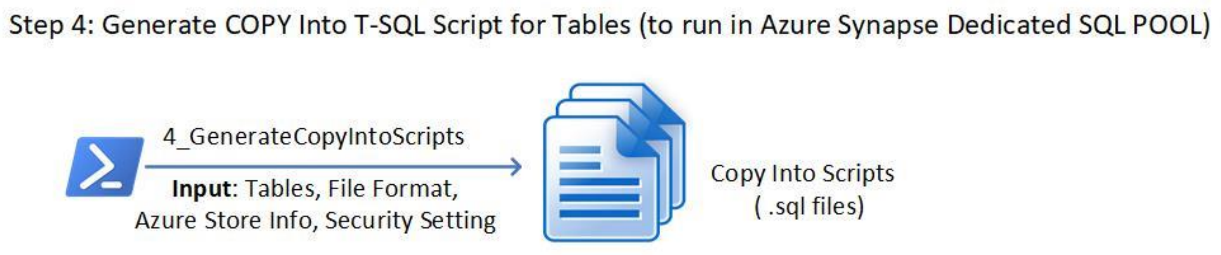
Figure 6 Step 3: Upload Data into Azure Storage

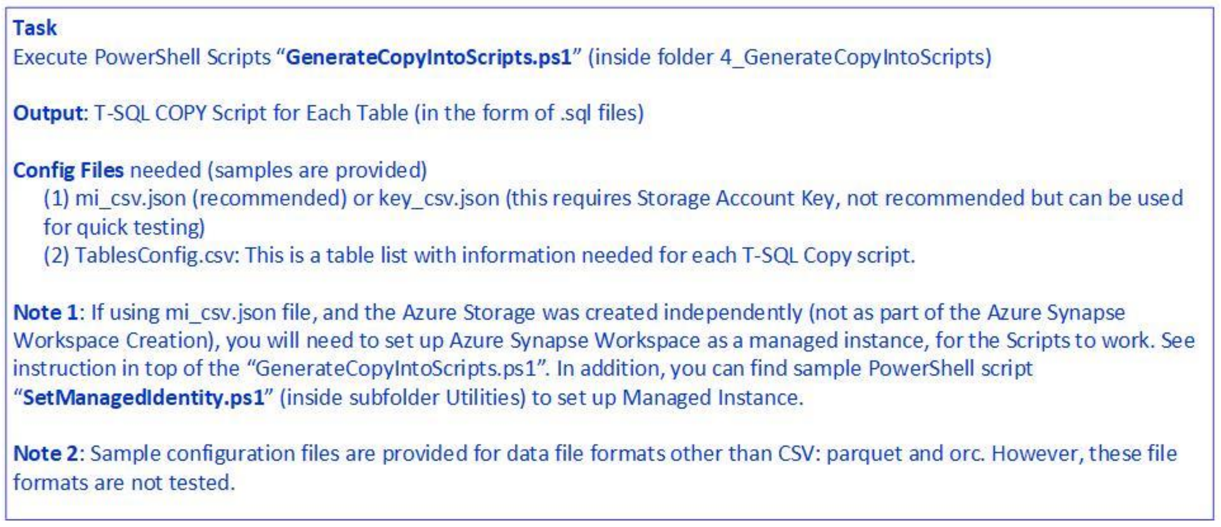
4.4 Step 4: Generate COPY T-SQL Scripts

The process of generating COPY T-SQL Scripts is illustrated in Figure 7. For details on how to run each module, please refer to Section 5, Detailed Reference Guide for Each Module.

After executing each of the generated T-SQL Copy Script in the format of .sql files, the corresponding data will be imported into Azure Synapse. The execution step is carried out in next step, Step 5.

This step only generates T-SQL Scripts, it does not execute any T-SQL Scripts.


Step 4: Generate COPY Into T-SQL Script for Tables (to run in Azure Synapse Dedicated SQL POOL)
4_GenerateCopyIntoScripts
Input: Tables, File Format, Azure Store Info, Security Setting
Copy Into Scripts (.sql files)


Task
Execute PowerShell Scripts "**GenerateCopyIntoScripts.ps1**" (inside folder 4_GenerateCopyIntoScripts)

Output: T-SQL COPY Script for Each Table (in the form of .sql files)

Config Files needed (samples are provided)
(1) mi_csv.json (recommended) or key_csv.json (this requires Storage Account Key, not recommended but can be used for quick testing)
(2) TablesConfig.csv: This is a table list with information needed for each T-SQL Copy script.

Note 1: If using mi_csv.json file, and the Azure Storage was created independently (not as part of the Azure Synapse Workspace Creation), you will need to set up Azure Synapse Workspace as a managed instance, for the Scripts to work. See instruction in top of the "GenerateCopyIntoScripts.ps1". In addition, you can find sample PowerShell script "**SetManagedIdentity.ps1**" (inside subfolder Utilities) to set up Managed Instance.

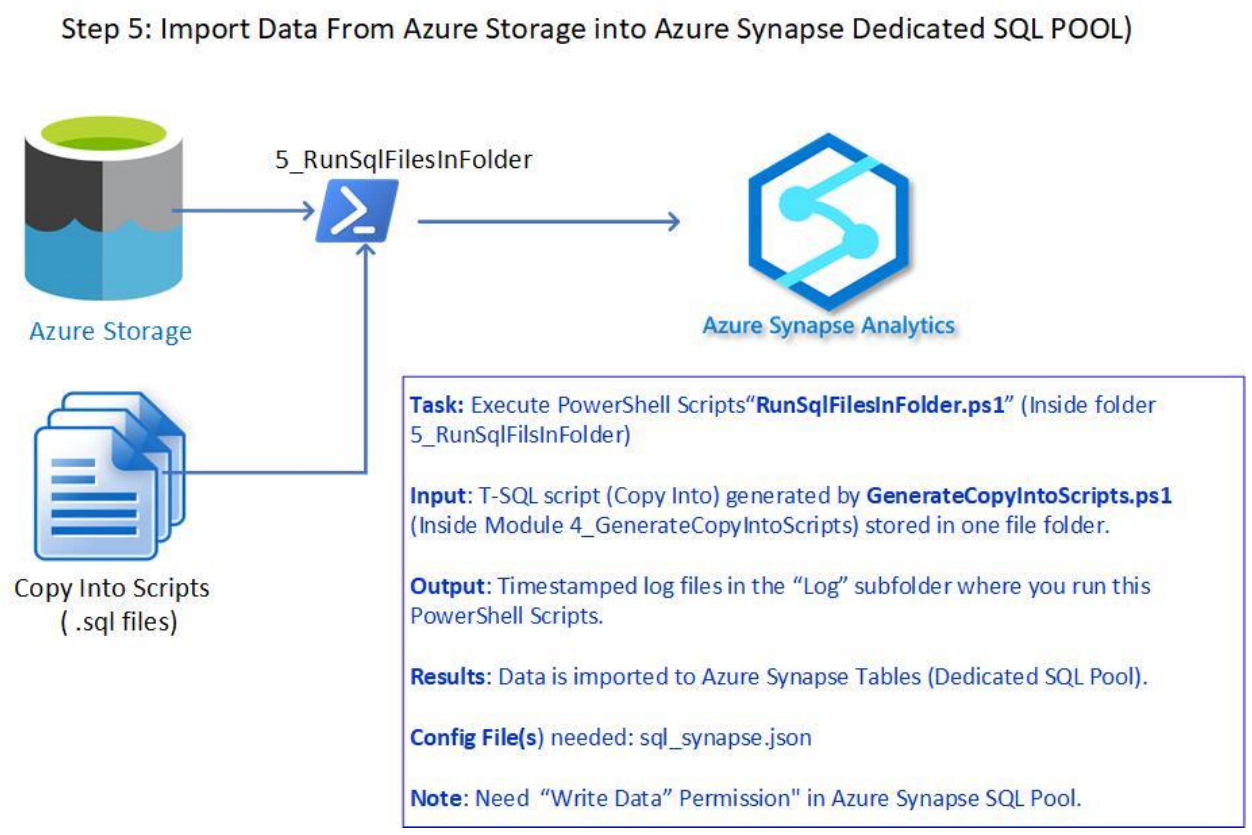
Note 2: Sample configuration files are provided for data file formats other than CSV: parquet and orc. However, these file formats are not tested.

Figure 7 Step 4: Generate T-SQL COPY Scripts

4.5 Step 5: Import Data into Azure Synapse (SQL Pool)

The process of generating COPY T-SQL Scripts is illustrated in Figure 8. For details on how to run each module, please refer to Section 5, Detailed Reference Guide for Each Module.

After this step, the data stored in Azure Storage will be imported into Azure Synapse SQL pool. The program also produces a log file, itemizing the results of each T-SQL Scripts. If there is errors, the errors will be found in the log file.


Step 5: Import Data From Azure Storage into Azure Synapse Dedicated SQL POOL)
5_RunSqlFilesInFolder
Azure Storage
Azure Synapse Analytics
Copy Into Scripts (.sql files)

Task: Execute PowerShell Scripts"**RunSqlFilesInFolder.ps1**" (Inside folder 5_RunSqlFilsInFolder)

Input: T-SQL script (Copy Into) generated by **GenerateCopyIntoScripts.ps1** (Inside Module 4_GenerateCopyIntoScripts) stored in one file folder.

Output: Timestamped log files in the "Log" subfolder where you run this PowerShell Scripts.

Results: Data is imported to Azure Synapse Tables (Dedicated SQL Pool).

Config File(s) needed: sql_synapse.json

Note: Need "Write Data" Permission" in Azure Synapse SQL Pool.

Figure 8 Step 5: Import Data into Azure Synapse SQL Pool

5 Detailed Reference Guide for Each Module

5.1 1_TranslateMetaData

Use Module **1_TranslateMetaData** to Translate SQL Server Tables into Azure Synapse Tables and save them into target .sql files. The folder name and files are illustrated in Figure 9. You don't need to change GetTableMetaDataData.sql or TranslateTables.ps1.

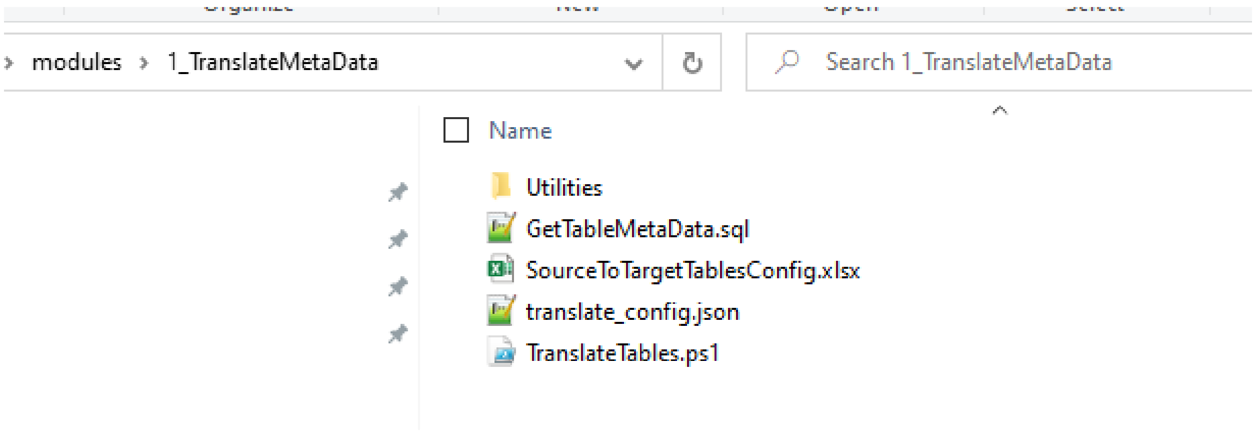

› modules › 1_TranslateMetaData
Search 1_TranslateMetaData
Name
Utilities
GetTableMetaData.sql
SourceToTargetTablesConfig.xlsx
translate_config.json
TranslateTables.ps1

Figure 9 Module 1: Module 1_TranslateMetaData Folder and Files

You need to run **TranslateMetaData.ps1** which will prompt you for the names of two configuration files:

(1) translate_config.json: This file specifies the SQL Server Name, Security setting, and Output files folder.
(2) SourceToTargetTablesConfig.xlsx. This file has a list of items that specify the SQL Server Database, Schema Name, Table Name, desired Synapse table Schema name, and table distribution.

The definition and sample values for each row in translate_config.json file is described in Table 1.

Table 1 Module 1: Translate Meta Data (translate_config.json)

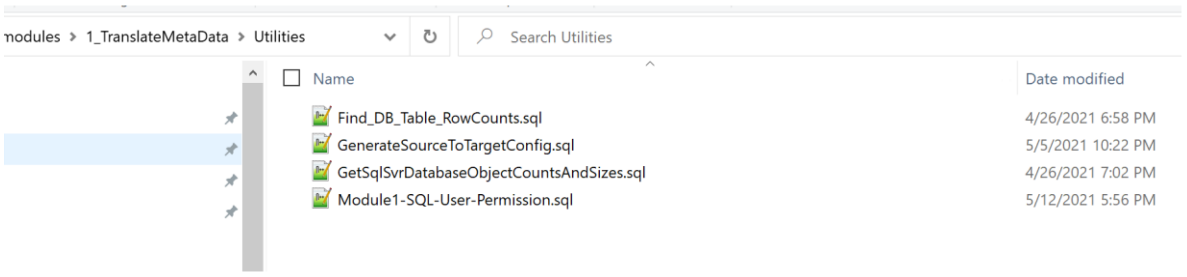
Parameter Name	Description	Values (Sample)
ServerName	Fully qualified SQL Server Name	.\\YourSQLServerName or YourFullyQualifiedSqlServerName
IntegratedSecurity	YES or NO for IntegratedSecurity	YES or NO
ThreePartsName	YES or NO for Three-Parts-Name code generation (db.schema.table)	YES or NO
OutputFolder	Full File Path where the translated code will be stored.	C:\\migratemaster\\output\\1_TranslateMetaData

The definition and sample values for each column in "SourceToTargetTablesConfig.xlsx" is described in Table 2.

Table 2 Module 1: Translate Meta Data (SourceToTargetTablesConfig.xlsx)

Parameter Name	Description	Values (Sample)
Active	1 – Run line, 0 – Skip line.	0 or 1
DatabaseName	Database Name	AdventureWorksDW2017
SchemaName	SQL Server Schema Name	dbo
AsaDatabaseName	Azure Synapse Database Name	SynapseSQLPool (This field is not currently used. It was planned for future use).
AsaSchemaName	The schema name to be used in Azure Synapse SQL pool	dbo_asa, edw
ObjectName	Table Name	DimEmployee
ObjectType	Type of the SQL Object (Table, View, Stored Procedure)	Table
DropFlag	YES or NO. If Yes, drop table statement will be generated for Create Table DDL.	YES or NO.
AsaTableType	Table types: Heap or CCI	HEAP, CCI
TableDistrubution	Azure Synapse Table Distribution Type (Round_Robin, Hash, Replicate).	Round_Robin, Replicate, Hash
HashKeys	Keys to be used as Hash keys. Defined this field only if the table is to be distributed as Hash.	ProductKey. If multiple keys, they must be separated by ",". See the sample configuration file "SourceToTargetConfig.xlsx" for more details.

In addition, in the subfolder named "Utilities", we provided utilities, as illustrated in Figure 10.



Name	Date modified
Find_DB_Table_RowCounts.sql	4/26/2021 6:58 PM
GenerateSourceToTargetConfig.sql	5/5/2021 10:22 PM
GetSqlSvrDatabaseObjectCountsAndSizes.sql	4/26/2021 7:02 PM
Module1-SQL-User-Permission.sql	5/12/2021 5:56 PM

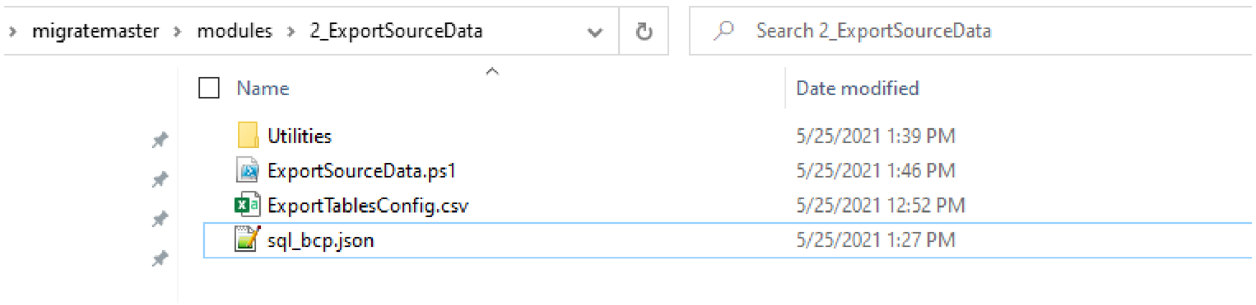
Figure 10 Utilities for Module 1: Translate Meta Data

- Find_DB_Table_RowCounts.sql – For each database connected, it returns the row count for each table.
- GenerateSourceToTargetConfig.sql – Run this against each SQL server database, it generates an initial starter configuration file that has the same structure as SourceToTargetTablesConfig.xlsx sample file. This will be to be used as input.

- GetSqlSvrDatabaseObjectCountsAndSizes.sql – For each dataset in the entire SQL server, it returns Database Size, #Tables, #Views, #Stored Procedures, #Triggers.
- Module1-SQL-User-Permission.sql – Sample T-SQL script to set up permission for the migration user(s),

5.2 2_ExportSourceData

Use Module **2_ExportSourceData** to export data for specified tables into .csv files. The folder name and files are illustrated in Figure 11.



migratemaster > modules > 2_ExportSourceData
Search 2_ExportSourceData
Name
Date modified

Name	Date modified
Utilities	5/25/2021 1:39 PM
ExportSourceData.ps1	5/25/2021 1:46 PM
ExportTablesConfig.csv	5/25/2021 12:52 PM
sql_bcp.json	5/25/2021 1:27 PM

Figure 11 Module 2: Export Source Data

You need to run **ExportSourceData.ps1** PowerShell script which will prompt you for the names of two configuration files:

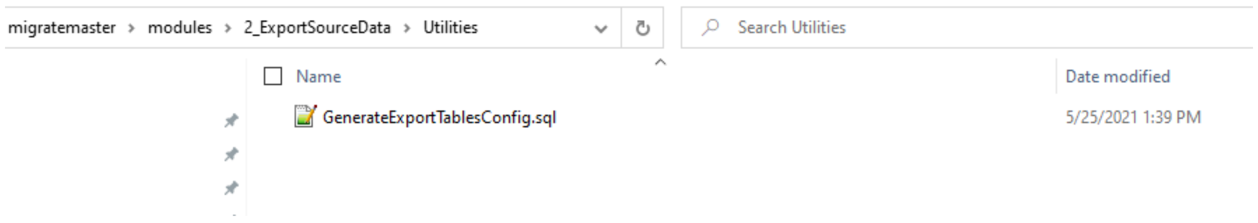
(1) sql_bcp.json: This simple file specifies the fully qualified SQL server name, type of security, and bcp installation directory. This utility comes with SQL server, you just need to locate where it is.
(2) ExportTablesCofig.csv: You can use "GenerateTablesConfig.sql.sql" (in Utilities folder) to automatically generate an initial configuration file with little needs of manual editing. A sample file is provided for you for reference.

The definition and sample values for each row of the sql_bcp.json file is described in Table 3.

Table 3 JSON Configuration File Parameter for Module 2: Export Source Data

Parameter Name	Description	Values (Sample)
ServerName	Fully qualified SQL Server Name	.\\YourSQLServerName or YourSqlServerName or Fully Qualified Server name
ServerType	Type of the Server: SQL	SQL
IntegratedSecurity	YES or NO for Integrated Security	YES, NO
OutputFolder	Folder Name output table data will be stored	C:\\migratemaster\\output\\2_ExportSourceData
OutputFileExtension	File extension for the table data output	.csv or .txt
BcpLocation	Location of the BCP utility installed.	C:\\Program Files\\Microsoft SQL Server\\Client SDK\\ODBC\\130\\Tools\\Binn

In the subfolder “Utilities”, there is a T-SQL script file “GenerateExportTablesConfig.sql” to help you to produce output you can copy and paste (with header) into a .csv file, save it as “ExportTablesConfig.csv”.

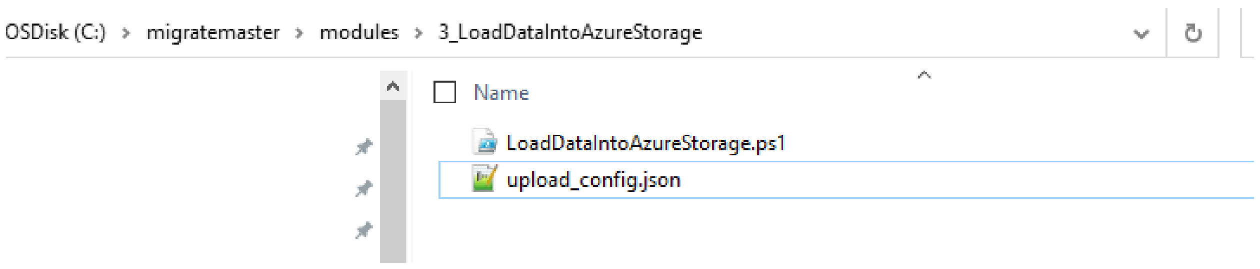


GenerateExportTablesConfig.sql 5/25/2021 1:39 PM

Figure 12 Utilities for Module 2: Export Source Data

5.3 3_LoadDataIntoAzureStorage

Use module **3_LoadDataIntoAzureStorage** to copy data from on-prem storage to Azure Storage.


OSDisk (C:) › migratemaster › modules › 3_LoadDataIntoAzureStorage
Name

LoadDataIntoAzureStorage.ps1
upload_config.json

Figure 13 Module 3: Load Data into Azure Storage (Azure Data Lake Store or Blob Storage)

You need to run **LoadDataIntoAzureStorage.ps1** which will prompt you for the names of one configuration file: upload_config.json.

Table 4 3_LoadDataIntoAzureStorage Config File (upload_config.json)

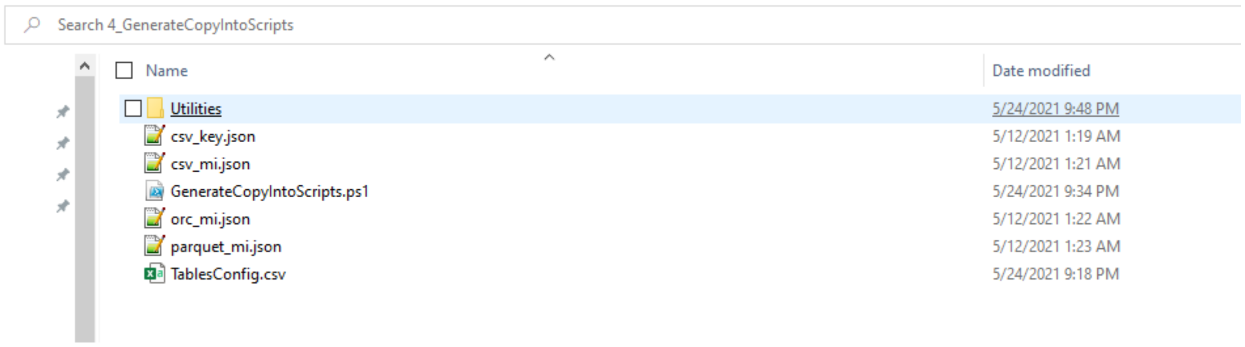
Parameter Name	Description	Values (Sample)
AzCopyPath	Location where AzCopy is installed	C:\\Z_Progs\\AzCopy10
AzureSubscriptionID	Your Azure Subscription ID	You can find out this value from Azure Portal if you have access to an Azure Subscription. It looks like this: abcxyz-123bn-bayn49gw-stuff-01234
AzureResourceGroup	Azure Resource Group for the Azure Storage where you'd like to upload files into.	From Azure portal you can find the resource group that contains the Azure Storage Account.
StorageAccountName	Azure Storage Account Name	From Azure portal you can your Azure Storage Account Name.
GenerateSASKey	Specify "Yes" or "Y" if you'd like a SAS key to be generated. If Yes, you will be prompted to login into your Azure account.	"Yes" or "Y". Any other values will be converted to "No".
SASKey	Provide your own SAS key if the answer to above "GenerateSASKey" is not Yes.	This is the SAS key generated by Azure. Very long string.
KeyExpirationTimeInHours	Specify how long the SAS key will be valid (in hours) if you'd like the SAS key be generated.	Positive Integer
ContainerName	Your Azure Storage Container Name where you'd like the files to be uploaded into.	
FoldersToUpload	Specify a list of local folders. One or more folders can be specified. Separated by comma.	["C:\\migratemaster\\output\\2_ExportSourceData\\Folder1", "C:\\migratemaster\\output\\2_ExportSourceData\\Folder2]

Below are some alternative methods to move data from your on-prem local storage to Azure Storage (Azure Data Lake Store or Blob Storage):

A. Azure Data Explorer - What is Azure Data Explorer? | Microsoft Docs
B. Use AzCopy - azcopy | Microsoft Docs
C. Azure Data Box Gateway - Microsoft Azure Data Box Gateway overview | Microsoft Docs
D. Azure Data Factory - Azure Data Factory Documentation - Azure Data Factory | Microsoft Docs

5.4 4_GenerateCopyIntoScripts

Use module **4_GenerateCopyIntoScripts** to generate ‘Copy Into’ T-SQL Scripts. The folder name and files are illustrated in Figure 14.


Search 4_GenerateCopyIntoScripts

Name	Date modified
Utilities	5/24/2021 9:48 PM
csv_key.json	5/12/2021 1:19 AM
csv_mi.json	5/12/2021 1:21 AM
GenerateCopyIntoScripts.ps1	5/24/2021 9:34 PM
orc_mi.json	5/12/2021 1:22 AM
parquet_mi.json	5/12/2021 1:23 AM
TablesConfig.csv	5/24/2021 9:18 PM

Figure 14 Module 4: Generate Copy Into T-SQL Scripts

You will need to run PowerShell Script **GenerateCopyIntoScripts.ps1** which will prompt you for the names of two configuration files:

(1) csv_mi.json or parquet_mi.json or orc_mi.json (for CSV, Parquet, or ORC respectively) that specifies parameters that are relevant to the T-SQL Command “Copy Into”.
(2) TablesConfig.csv that has the list of tables to be copied into Synapse from Azure Storage.

The definition and sample values for each column in JSON configuration files are defined in Table 5. Please note that we have provided various sample configuration files for different scenarios. It is coded in the file name.

Table 5 JSON Configuration File Data Fields for Module 4: Generate Copy Into T-SQL Scripts (adls_mi_csv.json)

Parameter Name	Description	Values (Sample)
StorateType	Specify type of Azure Storage. “blob” for Blob Storage, “adls” for Azure Data Lake Store.	blob or adls (case insensitive)
Credential	If providing Storage Account key, (IDENTITY= 'Storage Account Key', SECRET='replaceThisWithYourRealKey=='). If Managed Identity, (IDENTITY= 'Managed Identity')	(IDENTITY= 'Storage Account Key', SECRET='replaceThisWithYourRealKey==') or (IDENTITY= 'Managed Identity')
AccountName	Your Azure Storage Account Name (Blob or Data Lake Store). Get this from your Azure Storage Overview blade.	YourStorageAccountName
Container	Container Name in Azure Data Lake Store or Blob Storage.	migratemaster
RootFolder	Folder Under Container where the data will be uploaded. If it is blank (white space), data will be loaded under container	Folder1 or Blank (white space)

FileType	CSV or Parquet or orc	CSV or Parquet or orc
Compression	Compression algorithms used. CSV supports GZIP Parquet supports GZIP and Snappy ORC supports DefaultCodec and Snappy. Zlib is the default compression for ORC	GZIP, Snappy
FieldQuote	Identifier that is used for Field Quotations. Needed only if the FileType is CSV.	\"
FieldTerminator	Terminator used for your data fields (columns). Needed only if the FileType is CSV.	0x1F
RowTerminator	Row Terminator. Needed only if the FileType is CSV.	0x1E
Encoding	UTF8 is the only encoding supported at this time. Needed only if the FileType is CSV.	UTF8
MaxErrors	Maximum errors allowed for importing data. This should be an integer specifies the maximum number of reject rows allowed in the load before the COPY operation is canceled. Each row that cannot be imported by the COPY operation is ignored and counted as one error. If max_errors is not specified, the default is 0.	0, 100, 5000
ErrorsFolder	Folder name under the container where you'd like errors logged.	Errors
FirstRow	Line number of the data file to be used as first row. If no header, use 1, if there is one line as header, use 2. Needed only if the FileType is CSV.	1 or 2 or any #
SqlFilePath	Location where you'd like generated T-SQL Scripts to be stored.	C:\\migratemaster\\output\\4_GenerageCopyIntoScripts\\DfsMiCsv

The definition and sample values for each column in TablesConfig.csv is described in Table 6.

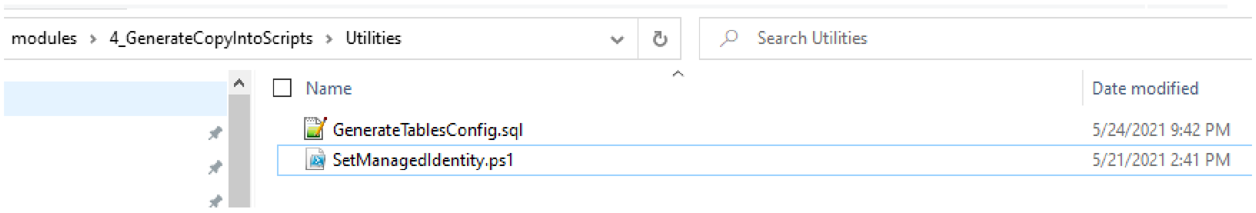
Table 6 CSV Configuration File Fields for Module 4: Generate Copy Into T-SQL Scripts (TablesConfig.csv)

Parameter Name	Description	Values (Sample)
Active	1 – Run line, 0 – Skip line.	0 or 1
DatabaseName	Database Name	AdventureWorksDW2017
SchemaName	SQL Server Schema Name	dbo
TableName	SQL Server Table Name	DimCustomer
IdentityInsert	On or OFF. "On" if the table contains Identity Data Field. "Off" if the table does not contain it. Leave it blank if the answer is no.	On or Off. If blank or any other values, it will be equivalent to "Off".

TruncateTable	Yes if table needs to be truncated before loading data into it.	Yes or No. Case Insensitive.
AsaDatabaseName	Azure Synapse Database Name (SQL Pool DB Name), this must match your actual DB name.	AsaDbName
AsaSchema	Azure Synapse Schema Name	dbo_asa, edw

In addition, there are two utilities that will help you to jump start the module4 code generation

(1) GenerateTablesConfig.sql: T-SQL Scripts that you can run against your SQL server database to produce a starter TablesConfig.csv file. You can change the script to include the actual Azure Synapse database name.
(2) SetManagedIdentity.ps1: PowerShell script to help you to set up managed identity. This step is not required if you created your Azure Storage when you created your Azure Synapse workspace or otherwise already set up.



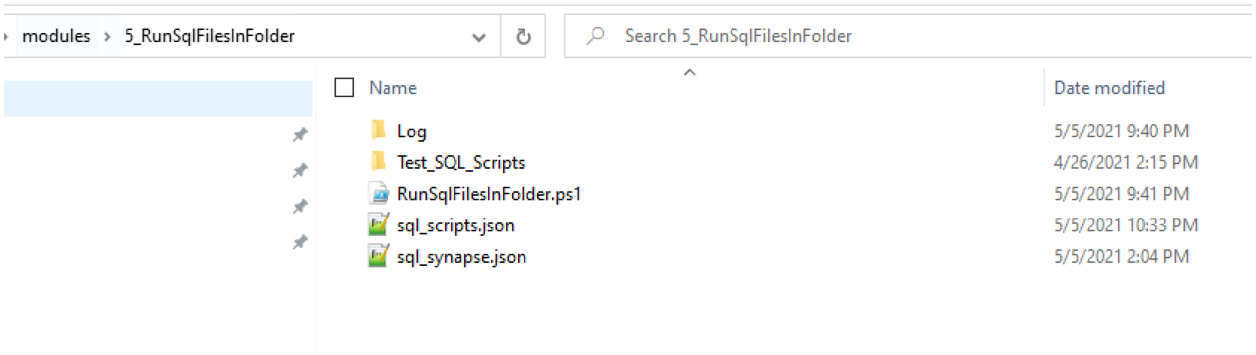
Name	Date modified
GenerateTablesConfig.sql	5/24/2021 9:42 PM
SetManagedIdentity.ps1	5/21/2021 2:41 PM

Figure 15 T-SQL and PowerShell Utilities to Jumpstart Module4

5.5 5_RunSqlScriptsInFolder – Create Azure Synapse Meta Data

Use module **5_RunSqlFilesinFolder** to create Azure Synapse objects specified in one specified folder containing all T-SQL Scripts. You can use this module to execute the T-SQL Scripts generated by Module 1_TranslateMetaData and Module 4_GenerateCopyIntoScripts.

The folder and files are illustrated in Figure 16.



Name	Date modified
Log	5/5/2021 9:40 PM
Test_SQL_Scripts	4/26/2021 2:15 PM
RunSqlFilesInFolder.ps1	5/5/2021 9:41 PM
sql_scripts.json	5/5/2021 10:33 PM
sql_synapse.json	5/5/2021 2:04 PM

Figure 16 Module 5: Run SQL Scripts in Folder - Execute T-SQL Scripts in one folder (for SQL Server or Azure Synapse)

You need to run RunSqlFilesInFolder.ps1 with only one configuration file: sql_scripts.json. For your convenience we provided a sample sql_synapse.json to guide you to create one for your own Azure Synapse instance.

Table 7 Module 5_RunSqlFilesInFolder (sql_scripts.json)

Parameter Name	Description	Values (Sample)
ServerName	Fully qualified SQL Server Name	.\\YourSQLServerName or yourSynapseWorkSpaceName.sql.azuresynapse.net
DatabaseName	Database Name	SQL
IntegratedSecurity	YES or NO for IntegratedSecurity	YES, NO. Use the value No if you are executing Azure Synapse T-SQL Scripts from your desktop.
SqlFilesFolder	The Folder Name where all the T-SQL Scripts are stored.	C:\\migratemaster\\output\\4_GenerateCopyIntoScripts\\AdlsMiCsv